DANIELLE DURSKI FIGUEIREDO

PROPOSTA DE UM ALGORITMO HÍBRIDO BASEADO EM EVOLUÇÃO DIFERENCIAL PARA OS PROBLEMAS DE *P*-MEDIANAS E DE MÁXIMA COBERTURA

Tese apresentada ao Programa de Pós-Graduação em Métodos Numéricos em Engenharia – área de concentração: Programação Matemática - dos setores de Tecnologia e de Ciências Exatas da Universidade Federal do Paraná, como requisito parcial à obtenção do grau de Doutora.

Orientadora: Luzia Vidal de Souza
Co-Orientador: Luiz Fernando Nunes

CURITIBA

2014

F475p Figueiredo, Danielle Durski
Proposta de um algoritmo híbrido baseado em evolução diferencial para os problemas de p-medianas e de máxima cobertura. / Danielle Durski Figueiredo.– Curitiba, 2014
98f. : il. [algumas color.] ; 30 cm.

Tese (doutorado) - Universidade Federal do Paraná, Setor de Tecnologia e Ciências Exatas, Programa de Pós-graduação em Métodos Numéricos em Engenharia, 2014.
Orientadora: Luiza Vidal de Souza -- Coorientador: Luiz Fernando Nunes.
Bibliografia: p. 84-90.

1. Otimização Combinatória. 2. Algoritmos Heurísticos. I. Universidade Federal do Paraná. II. Souza, Luiza Vidal de. III. Nunes, Luiz Fernando IV. Título.

CDD: 519.64

TERMO DE APROVAÇÃO

DANIELLE DURSKI FIGUEIREDO

PROPOSTA DE UM ALGORITMO HÍBRIDO BASEADO EM EVOLUÇÃO DIFERENCIAL PARA OS PROBLEMAS DE P-MEDIANAS E MÁXIMA COBERTURA.

Tese aprovada como requisito parcial para obtenção do grau de doutora no Programa de Pós-Graduação em Métodos Numéricos em Engenharia, da Universidade Federal do Paraná, pela seguinte banca examinadora:

Prof.ª Dr.ª Luzia Vidal de Souza.
Orientadora – Membro do PPGMNE/UFPR.

Prof. Dr. Leandro Magatão.
Membro da UTFPR.

Prof. Dr. Sérgio Fernando Mayerle.
Membro da UFSC.

Prof.ª Dr.ª Deise Maria Bertholdi Costa.
Membro do PPGMNE/UFPR.

Prof.ª Dr.ª Maria Teresinha Arns Steiner.
Membro do PPGMNE/UFPR e da PUCPR.

Curitiba, 29 de agosto de 2014.

Dedicatória

Dedico esse trabalho ao meu Pai, Erni, que sempre apoia e motiva a minha caminhada na vida, à minha Mãe Silvia (*in memoriam)*, que me cerca da sua luz, ao meu querido marido por estar sempre presente afetivamente quando preciso, e ao meu filho Augusto que enche de alegria e amor os meus dias.

AGRADECIMENTOS

À minha orientadora, Profa. Dra. Luzia Vidal de Souza, pelo acompanhamento, orientação e amizade.

Ao meu co-orientador, Prof. Dr. Luiz Fernando Nunes, pelas contribuições e sugestões no trabalho e amizade.

À Profa. Dra. Angela Olandoski Barboza, pelas valiosas sugestões no desenvolvimento deste trabalho e amizade.

Agradeço a todos os professores com quem tive contato, seja em sala de aula ou numa simples conversa sobre a pesquisa.

Ao Departamento Acadêmico de Matemática da UTFPR, pelo apoio e aprovação do afastamento, que foi fundamental para a conclusão deste trabalho.

Aos funcionários Maristela Bandil e Jair Anjos, por estarem sempre prontos a cooperar.

RESUMO

O estudo dos problemas de localização de instalações se relaciona diretamente com problemas organizacionais da sociedade, como por exemplo, a localização de escolas, postos de saúde, etc. Na sua forma geral, os problemas de *P*-Medianas e Máxima Cobertura são *NP-hard* e nas suas resoluções são utilizados métodos heurísticos. Os algoritmos de Evolução Diferencial (ED) são poderosos algoritmos de otimização evolucionária, propostos inicialmente, para problemas em espaços contínuos. Recentemente, têm sido propostas adaptações ao seu mecanismo de mutação diferencial para aplicação em problemas combinatórios. Este trabalho apresenta um novo algoritmo híbrido, utilizando algoritmos Evolução Diferencial e Busca Tabu, para a abordagem de problemas de *P*-Medianas e Máxima Cobertura. Introduz-se no operador de mutação diferencial de um algoritmo de Evolução Diferencial, o algoritmo Busca Tabu, com adaptações, a fim de que o mesmo possa ser aplicado para resolver problemas em um espaço de busca discreto. Testes computacionais foram realizados, com instâncias disponíveis na literatura, e comparados com outras meta-heurísticas e soluções ótimas obtidas com um modelo matemático. Os resultados encontrados sugerem que a técnica proposta é promissora e apropriada para a resolução dos problemas abordados, pois obteve-se na maioria dos testes soluções iguais ou melhores que alguns métodos presentes na literatura em tempos computacionais aceitáveis.

Palavras Chave: Otimização Combinatória, Algoritmos Heurísticos, Localização de Instalações.

ABSTRACT

The study of facility location problems is directly related to organizational problems of society, such as the location of schools, health centers , etc. . In its general form, the problem of *P*-Medians and Maximum Coverage is *NP*-hard, and heuristic methods are used to solve them. The Differential Evolution (DE) algorithms are powerful evolutionary optimization algorithms, originally proposed for problems in continuous spaces. Recently, it has been proposed adjustments that can be made to the mechanism of differential mutation for its application to combinational problems. This paper presents a new hybrid algorithm, using Differential Evolution Algorithms and Tabu Search, to address problems of *P*-Medians and Maximum Coverage. Be introduced to the operator of a differential mutation algorithm Differential Evolution, the Tabu Search algorithm with adaptations, so that it can be applied to solve problems in a discrete search space. Computational tests were performed, with instances available in the literature, and compared with other meta-heuristics and optimal solutions obtained from a mathematical model. The results suggest that the proposed technique is promising and appropriate for the resolution of the problems addressed, as was obtained in most testing solutions equal or better than some methods from the literature in acceptable computational time.

Keywords : Combinatorial Optimization, Heuristic Algorithms, Location of Facilities.

LISTA DE FIGURAS

FIGURA 1 – SOLUÇÃO INICIAL …….. 37
FIGURA 2 – ÓTIMO LOCAL ……. 37
FIGURA 3 – ELEMENTOS TABU …… 38
FIGURA 4 – ÓTIMO GLOBAL …….. 38
FIGURA 5 – ALGORITMO EVOLUTIVO BÁSICO 42
FIGURA 6 – ETAPAS DE EVOLUÇÃO DIFERENCIAL TRADICIONAL ……... 44
FIGURA 7 – PROCESSO DE GERAÇÃO DO VETOR MUTANTE (ED1) 46
FIGURA 8 – EXEMPLO DO PROCESSO DE CRUZAMENTO BINOMIAL 48
FIGURA 9 – EXEMPLO DO PROCESSO DE CRUZAMENTO EXPONENCIAL 49
FIGURA 10 – PROCESSO DE GERAÇÃO DE UMA COMPONENTE DO VETOR DOADOR (V_{doador}) 60
FIGURA 11– FLUXOGRAMA EDBT 62

LISTA DE QUADROS

QUADRO 1 – PROCEDIMENTO BUSCA TABU..... 40
QUADRO 2 – DESCRIÇÃO DAS VARIÁVEIS E COMPONENTES DO EDBT..... 57

LISTA DE TABELAS

TABELA 1 – COORDENADAS CARTESIANAS 63

TABELA 2 – MATRIZ DE DISTÂNCIAS …….. 63

TABELA 3 – RESULTADOS DOS TESTES COMPUTACIONAIS EM RELAÇÃO A LORENA *ET AL*. (2001) PARA PROBLEMAS DE *P*-MEDIANAS 70

TABELA 4 – RESULTADOS DOS TESTES COMPUTACIONAIS EM RELAÇÃO A VASCONCELOS *ET AL*. (2010) PARA PROBLEMAS DE *P*-MEDIANAS ONDE *N* = 324..... 74

TABELA 5 – RESULTADOS DOS TESTES COMPUTACIONAIS EM RELAÇÃO A VASCONCELOS *ET AL*. (2010) PARA PROBLEMAS DE *P*-MEDIANAS ONDE *N* = 818 74

TABELA 6 – RESULTADOS DOS TESTES COMPUTACIONAIS COMPARADOS COM TEITZ & BART PARA INSTÂNCIAS GERADAS PARA PROBLEMAS DE *P*-MEDIANAS..... 75

TABELA 7 – RESULTADOS DOS TESTES COMPUTACIONAIS COMPARADOS COM O *SOFTWARE* LINGO PARA PROBLEMAS DE MÁXIMA COBERTURA (*S* = 150M) 78

TABELA 8 – PARÂMETROS PARA AS INSTÂNCIAS LORENA_*P*-MEDIANAS 91

TABELA 9 – PARÂMETROS PARA AS INSTÂNCIAS GERADAS (*P*-MEDIANAS) 92

TABELA 10 – PARÂMETROS PARA AS INSTÂNCIAS LORENA_MÁXCOBERTURA COM *S* = 150 METROS 94

TABELA 11 – PARÂMETROS PARA AS INSTÂNCIAS LORENA_MÁXCOBERTURA COM *S* IGUAL A 800, 1200 E 1600 METROS 94

TABELA 12 – TEMPOS COMPUTACIONAIS DO *SOFTWARE* LINGO PARA INSTÂNCIAS GERADAS 95

TABELA 13 – RESULTADOS DOS TESTES COMPUTACIONAIS COMPARADOS COM A LANGRANGEANA/*SURROGATE* PARA PROBLEMAS DE MÁXIMA COBERTURA (S = 800, 1200, 1600 METROS)..... 96

LISTA DE GRÁFICOS

GRÁFICO 1 – ESPAÇO DE BUSCA PARA A OPERAÇÃO DE MUTAÇÃO DO EDBT 71
GRÁFICO 2 – DESEMPENHO DE CONVERGÊNCIA DO EDBT (*N* = 324) 72
GRÁFICO 3 – DESEMPENHO DE CONVERGÊNCIA DO EDBT (*N* = 818) 73
GRÁFICO 4 – TEMPOS COMPUTACIONAIS PARA AS INSTÂNCIAS LORENA_MÁXCOBERTURA 80

LISTA DE SIGLAS

Aes	—	Algoritmos Evolutivos
AG	—	Algoritmo Genético
AG_BL	—	Algoritmo Genético com busca Local Adaptativo
EDBT	—	Algoritmo Híbrido Proposto (Evolução Diferencial e Busca Tabu)
BT	—	Busca Tabu
CPU	—	*Central Processing Unit*
ED	—	Evolução Diferencial
EDD	—	Evolução Diferencial Discreta
FIFO	—	*First In First Out*
GRASP	—	*Greedy Randomized Adaptive Search Procedure*
JFO	—	Jumping Frog Optimization (Algoritmo de Otimização por Saltos de Rãs)
MPI	—	Método de Ponto Interior
PMC	—	Problema de Máxima Cobertura
PPM	—	Problema de *P*-Medianas
SAMU – SP	—	Serviço de Atendimento Móvel Urgente do Estado de São Paulo
SIGs	—	Sistemas de Informações Geográficas

SUMÁRIO

1 INTRODUÇÃO 15
1.1 OBJETIVOS 17
1.1.1 Objetivo Geral 17
1.1.2 Objetivos Específicos 17
1.2 CONTRIBUIÇÃO CIENTÍFICA 17
1.3 LIMITAÇÕES DO TRABALHO 18
1.4 ORGANIZAÇÃO DO TRABALHO 18
2 REVISÃO DE LITERATURA 20
2.1 PROBLEMAS DE LOCALIZAÇÃO DE INSTALAÇÕES 20
2.1.1 Histórico 20
2.1.2 Problema de *P*-Medianas 23
2.1.3 Problema de Máxima Cobertura 25
2.2 MÉTODOS DE SOLUÇÃO PARA PROBLEMAS DE LOCALIZAÇÃO 26
2.2.1 Métodos Exatos 27
2.2.2 Métodos Heurísticos 28
2.2.3 Meta-heurísticas 30
2.2.4 Métodos Híbridos 31
2.3 TEITZ & BART 34
2.3.1 Descrição do algoritmo Teitz & Bart 35
2.4 BUSCA TABU 36
2.4.1 Implementação do algoritmo Busca Tabu 39
2.5 ALGORITMOS DE EVOLUÇÃO 41
2.6 EVOLUÇÃO DIFERENCIAL 42
2.6.1 Inicialização do algoritmo Evolução Diferencial 43
2.6.1.1 Avaliação dos indivíduos 44
2.6.1.2 População Inicial 45
2.6.1.3 Operador de mutação 45
2.6.1.4 Cruzamento (*Crossover)* 47
2.6.1.5 Seleção 49
2.6.1.6 Parâmetros 49

2.6.1.7 Critério de término 50
2.6.2 Estratégias de Evolução Diferencial 50
2.6.3 Evolução Diferencial Discreta 52
3. METODOLOGIA 56
3.1 ALGORITMO HÍBRIDO PROPOSTO 56
3.4.1 Exemplo Ilustrativo para um PPM..... 62
4 TESTES COMPUTACIONAIS E ANÁLISE DOS RESULTADOS..... 68
4.1 TESTES E RESULTADOS PARA OS PPM..... 69
4.2 TESTES E RESULTADOS PARA OS PMC 77
5 CONCLUSÕES 81
5.1 TRABALHOS FUTUROS 83
REFERÊNCIAS 84
ANEXO 1 91
ANEXO 2 95
ANEXO 3 96

1 INTRODUÇÃO

Uma das mais importantes decisões em uma organização pública ou privada é a definição da localização de instalações, devido aos elevados investimentos envolvidos.

Os problemas de localização de instalações têm, geralmente, como objetivo servir ou suprir a população de uma dada área geográfica a partir de centros de distribuição. O que se busca determinar nestes problemas é a quantidade e a localização de instalações que possam atender, de forma mais econômica, um conjunto de clientes. A escolha de onde localizar as instalações deve ser feita de modo a otimizar uma função objetivo bem definida, obedecendo às restrições do problema específico considerado.

As aplicações são geralmente divididas entre: setores públicos e privados. No caso de setor público, aplicações maximizam a satisfação dos clientes em detrimento dos custos necessários para o alcance de tal objetivo, uma vez que em geral, os custos não são estimados com exatidão. Entre os exemplos de aplicações em setores públicos estão a localização de: escolas, postos de saúde, corpo de bombeiros, ambulâncias, viaturas de polícia, pontos de ônibus, entre outros. No caso do setor privado, custos chamados fixos estão envolvidos, e suas aplicações envolvem em geral fábricas, depósitos, torres de transmissão, lojas de franquias, entre outros [Ribeiro, 2008].

De acordo com Crainic e Laporte (1997), problemas de localização são classificados em: Problemas de Cobertura, Problemas Centrais e Problemas de *P*-Medianas.

Problemas de localização de instalações são considerados como problemas *NP-hard* [Garey e Jonhson, 1979]. Quando problemas de otimização, classificados como *NP-hard* são abordados com métodos exatos, dificuldades podem ser encontradas, especialmente nos casos de dimensão elevada, uma vez que o tempo computacional necessário para a obtenção do valor ótimo global cresce exponencialmente à medida que se aumentam os dados de entrada. Esta dificuldade faz com que os pesquisadores busquem outros métodos para solução destes problemas. Entre estes métodos estão as heurísticas e as meta-heurísticas.

Maranzana (1964) e Teitz e Bart (1968) apresentam dois dos primeiros métodos heurísticos, propostos na literatura, para resolução do Problema das *P*-Medianas. Desde então, diversos métodos heurísticos têm sido desenvolvidos para Problemas de Localização.

Meta-heurísticas são procedimentos destinados a encontrar uma solução adequada, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual deve ser modelada para cada problema específico. Conforme Chaves (2003), a principal característica das meta-heurísticas é a capacidade que estas possuem de escapar de ótimos locais dando certa flexibilidade às restrições da função objetivo.

Dentre as meta-heurísticas encontram-se os algoritmos evolutivos que fazem parte da computação evolutiva. Algoritmos evolutivos foram desenvolvidos com inspiração no processo biológico de evolução. A família de algoritmos evolutivos apresenta um grande número de técnicas de otimização e métodos heurísticos aplicados em problemas de diferentes áreas de conhecimento, principalmente em ciências e engenharia. Nas últimas décadas, novos métodos evolutivos surgiram, como por exemplo, os Algoritmos de Evolução Diferencial.

Os algoritmos de Evolução Diferencial são algoritmos robustos de otimização evolucionária, propostos inicialmente na década de 1990, para a otimização de sistemas com variáveis contínuas. Recentemente, adaptações têm sido propostas ao seu mecanismo de mutação diferencial para otimização de problemas combinatórios.

São muitas as propostas encontradas na literatura com o objetivo de melhorar a eficiência de meta-heurísticas, procurando adicionar buscadores locais ou fazer combinações com outras técnicas de resolução. Estas abordagens são comumente denominadas de hibridização e não existe uma regra principal que determina como são feitas estas adições ou combinações.

A implementação de propostas de adaptação e hibridização dos algoritmos evolutivos, clássicos da literatura, como por exemplo, os algoritmos Genéticos, Evolução Diferencial, etc., são propostas e procuram obter maior eficiência na busca por soluções.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo geral da presente pesquisa é apresentar um novo algoritmo híbrido, utilizando algoritmos de Evolução Diferencial e Busca Tabu, para a abordagem dos seguintes Problemas de Localização: *P*-Medianas e Máxima Cobertura.

1.1.2 Objetivos Específicos

Este trabalho tem os seguintes objetivos específicos:

i. apresentar uma proposta inédita, de adaptação na operação de mutação diferencial em um algoritmo de Evolução Diferencial para aplicação em problemas de otimização discreta, especificamente, Problemas de *P*-Medianas e Máxima Cobertura;

ii. propor, após testes computacionais, regras determinísticas para alguns parâmetros do algoritmo proposto neste trabalho a fim de diminuir a interferência do usuário na execução do mesmo;

iii. analisar o comportamento das soluções comparando-as com instâncias presentes na literatura e geradas aleatoriamente.

1.2 CONTRIBUIÇÃO CIENTÍFICA

O presente trabalho busca trazer uma contribuição para a área de Pesquisa Operacional, no que diz respeito à otimização das soluções para o Problema de *P*-Medianas e Máxima Cobertura, através de um processo de hibridização dos algoritmos Evolução Diferencial e Busca Tabu, apresentando ainda, para o algoritmo Evolução Diferencial, uma proposta inédita de adaptação para tais problemas na

operação de mutação diferencial, tendo em vista que o mesmo, na sua forma clássica, aborda somente problemas com soluções no domínio contínuo.

1.3 LIMITAÇÕES DO TRABALHO

Para Problemas de *P*-Medianas, este trabalho está limitado ao modelo não capacitado, onde cada instalação pode atender todos os pontos de demanda próximos a ela como se tivesse capacidade infinita.

Dificuldades surgiram em se encontrar instâncias que forneçam as coordenadas cartesianas dos vértices para a realização e validação da metodologia utilizada. Desta forma, foi necessário gerar instâncias para a verificação e análise dos resultados.

Quanto ao tamanho das instâncias geradas, os algoritmos são testados em instâncias que variam de 100 até um máximo de 600 nós e a distribuição destes pontos é uniforme.

As distâncias entre os pontos são calculadas utilizando a métrica euclidiana para o algoritmo testado, distanciando a aplicação da metodologia de problemas reais.

1.4 ORGANIZAÇÃO DO TRABALHO

O presente trabalho está estruturado em cinco capítulos. Este primeiro capítulo apresenta a introdução, os objetivos: geral e específicos, a contribuição científica e as limitações do trabalho. Os capítulos subsequentes deste trabalho estão organizados como segue.

No segundo capítulo é apresentada uma revisão bibliográfica sobre o Problema de Localização de Instalações. Os modelos matemáticos e os trabalhos que abordam os problemas da localização de *P*-Medianas e Máxima Cobertura são apresentados. Neste capítulo encontram-se as meta-heurísticas Teitz & Bart, Busca Tabu e Evolução Diferencial, com suas características específicas. Essas meta-heurísticas são utilizadas na comparação de resultados e construção do algoritmo híbrido proposto.

No terceiro capítulo encontra-se a metodologia adotada neste trabalho, que apresenta a proposta inédita de adaptação no operador de mutação diferencial do algoritmo Evolução Diferencial com o propósito de que o mesmo atue em problemas de *P*-Medianas e de Máxima Cobertura.

No quarto capítulo encontram-se os resultados dos testes computacionais executados e comparados com as soluções ótimas obtidas para o modelo matemático apresentado na literatura e de outros métodos da literatura.

As conclusões finais e recomendações para trabalhos futuros encontram-se no quinto capítulo.

2 REVISÃO DE LITERATURA

2.1 PROBLEMAS DE LOCALIZAÇÃO DE INSTALAÇÕES

Problemas de localização e suas aplicações constituem uma importante linha de pesquisa em Otimização Combinatória e despertam interesse nas mais diferentes áreas de conhecimento, tais como a Economia, a Administração, a Pesquisa Operacional, a Geofísica e a Engenharia. Tais problemas buscam a localização ideal para instalar facilidades com o objetivo de atender a uma demanda minimizando os custos deste processo.

Neste trabalho, os problemas de localização de instalações são abordados mediante um breve histórico. Em seguida, os modelos matemáticos para os problemas de localização de medianas e de máxima cobertura, são apresentados de forma mais aprofundada, por serem o foco de estudo deste trabalho.

2.1.1 Histórico

Os Problemas de Localização são estudados desde o século XVII, quando Pierre de Fermat buscava solucionar o seguinte problema: dados três pontos de um plano (vértices de um triângulo), encontrar o ponto do plano, tal que a soma das distâncias entre cada um dos vértices e este ponto seja mínima [Reese, 2006]. Já Alfred Weber, no século XX, desenvolveu o modelo que deu origem à teoria das localizações. Seu trabalho abordou a localização de uma indústria, considerando os fornecedores e os consumidores e buscou a melhor localização de forma a proporcionar o menor custo em termos de transporte. Nesse trabalho, Weber considerou cada um dos três pontos correspondendo a um cliente e buscou encontrar o referido ponto equivalendo a encontrar a melhor localização de uma instalação para atender seus clientes [Azzoni, 1982].

Horner (2009) afirma que o trabalho de Weber e outros da época tinham abordagem contínua, uma vez que para determinar uma localização no plano, o espaço de solução é infinito. No entanto, nas diversas aplicações em regiões

urbanas, observou-se que o espaço solução é discreto, sendo necessário desenvolver estudos para uma abordagem discreta.

Na década de 60, com o desenvolvimento da teoria dos grafos, iniciou-se o estudo de localização em redes, onde Hakimi (1965) mostrou que o espaço de soluções consiste apenas nos vértices do grafo, e as distâncias são medidas ao longo dos arcos. Propriedades matemáticas relevantes da teoria de grafos facilitam a modelagem, tornando possível o desenvolvimento de procedimentos eficientes para resolvê-los. Tais problemas são modelados através da Programação Matemática Inteira, uma das técnicas utilizadas da Pesquisa Operacional para problemas de otimização. Os problemas de otimização são aqueles que buscam uma solução denominada ótima, que atenda a um determinado objetivo a ser otimizado, chamado de função objetivo, que em geral, corresponde a minimizar custos ou maximizar lucros e ainda atenda às restrições do modelo. Os problemas de Programação Inteira têm como característica principal, a impossibilidade das variáveis assumirem valores contínuos, somente discretos (inteiros). Parte desses problemas são classificados com Problemas de Otimização Combinatória.

Dessa forma, podemos definir os Problemas de Localização como Problemas de Otimização Combinatória, que têm como objetivo a tomada de decisão da localização de novas instalações em uma rede que possua pontos de demanda a serem atendidos, de acordo com uma função objetivo a ser otimizada. Diversas são as aplicações dos problemas de localização, tais como a localização de depósitos, escolas, centros de saúde, empresas, postos policiais, supermercados, antenas de telefonia, terminais de integração de ônibus, estações de tratamento de água, roteadores em redes de computadores, entre outros. Os modelos de localização têm sido estudados exaustivamente e essas pesquisas podem ser acompanhadas na literatura

Segundo Crainic e Laporte (1997), os principais modelos de localização estão assim classificados:

- Modelos de cobertura: O objetivo é minimizar o custo de localização de uma instalação pela qual um máximo nível de cobertura é obtido. Os problemas de cobertura estão divididos em *Location Set Covering Problem* (Problema de Cobertura de Conjuntos) e *Maximal Covering Problem* (Problemas de Máxima Cobertura).

- Modelos centrais: O objetivo é localizar *p* instalações em uma rede, minimizando a máxima distância entre os vértices e as instalações, ou entre um nó de origem e a instalação mais próxima. Esse problema é conhecido como *P–Center Problem*, ou Problema de Minimax.

- Modelos medianos: O objetivo é localizar *p* instalações nos vértices de uma rede e alocar demandas nessas facilidades de forma a minimizar a soma das distâncias entre as facilidades e os pontos de demanda do consumidor. Segundo Crainic e Laporte (1997), se a instalação é não capacitada e o número de instalações é fixo, tem-se um Problema de *P*-Medianas.

Segundo Dubke (2006), alguns autores descrevem uma taxonomia que amplia os problemas de localização de instalações em:

- Modelos planos (planar) e modelos de rede (*network model*): Nos modelos planos, a demanda ocorre em qualquer lugar no plano (com coordenadas *X* e *Y*). Nesses modelos, supõe-se a inexistência de restrições de percurso, de modo que se pode usar a distância mais curta. Nos modelos de rede, assume-se que as instalações e os pontos de demanda estão localizados nos nós da rede;
- Modelos contínuos e discretos: Quando o modelo permite que as instalações sejam localizadas em qualquer lugar dentro de um particular espaço de soluções (no plano), dá-se a denominação de modelo contínuo. Em contraste aos modelos contínuos, os modelos de localização em rede são classificados como modelos discretos, pois se assume que a demanda e as instalações estão localizadas nos nós de uma rede, em um conjunto finito de localizações;
- Modelos estáticos e dinâmicos: A maioria dos modelos de localização conhecidos é estática, ou seja, independem do tempo. Mas, existem os modelos dinâmicos, quando se considera a questão de onde e quando localizar, ou seja, a condição tempo é considerada;
- Modelos probabilísticos (estocásticos) e determinísticos: Como os modelos podem ser estáticos e dinâmicos, estes também podem ser probabilísticos, quando são sujeitos a incertezas ou determinísticos, quando não sujeitos a incertezas;
- Modelos para um único produto (*single product*) ou modelos que levam em consideração múltiplos produtos (*multi-products* ou *multicommodities*);

- Modelos com um único objetivo, que pode ser a determinação de mínimo custo ou a localização ou com múltiplos objetivos com a determinação de mínimo custo e a maximização da demanda coberta;
- Modelos de localização capacitados ou não capacitados: Os modelos de localização (cobertura, medianos e centrais) tratam a localização como um problema não capacitado ou com capacidade ilimitada (*uncapacitated location model*). Entretanto, existem modelos que já impõem esse limite ou tamanho da capacidade nas restrições ao modelo (*capacitated facility location model*).

Existem, entretanto, inúmeros fatores que têm interferido na decisão da localização de instalações, além de fatores referentes unicamente à minimização de distância e custo. Dubke (2006) cita alguns fatores qualitativos que interferem na decisão da localização de instalações:

- Custos de instalação, operação e transporte;
- Localização dos mercados fornecedores e consumidores;
- Disponibilidade e custos de mão de obra, serviços de comunicação, saúde, energia e segurança;
- Taxa de câmbio e barreiras comerciais;
- Regulamentação de impacto ambiental;
- Grau de organização sindical;
- Disponibilidade e custos de serviços públicos;
- Facilidades para o sistema de transporte;
- Localização dos concorrentes;
- Incentivos governamentais.

Dentre os problemas de localização de instalações, os que foram utilizados neste trabalho foram os Problemas de *P*-Medianas e Máxima Cobertura, que estão descritos a seguir.

2.1.2 Problema de *P*-Medianas

O Problema de *P*-Medianas (PPM) é um problema de localização e também um problema de alocação, que tem como objetivo, minimizar a soma dos custos de

distribuição entre instalações, que fornecem um serviço e pontos de demanda que precisam ser atendidos. Em função disso, estes problemas são também chamados de *Minisum*. As formulações mais importantes desses problemas foram apresentadas por Hakimi (1964). Este provou que ao menos um conjunto de pontos ótimos do problema será constituído de vértices do grafo, e sua importância deve-se ao fato de fazer com que os métodos de busca não percam tempo procurando a solução sobre os arcos do grafo.

Segundo Christofides (1975), o PPM pode ser formulado como um problema de programação linear inteira binária, da seguinte forma:

$$Z = \min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}\, x_{ij} \,. \tag{1}$$

sujeito a: $\sum_{i=1}^{n} x_{ij} = 1 \qquad j \in N \,.$ (2)

$$\sum_{i=1}^{n} x_{ii} = p. \tag{3}$$

$$x_{ij} \leq x_{ii} \qquad i, j \in N \,. \tag{4}$$

$$x_{ij} \in \{0,1\} \qquad i, j \in N \,. \tag{5}$$

Onde: $\left[d_{ij} \right]_{n \times n}$ é uma matriz simétrica de distâncias ponderadas; $\left[x_{ij} \right]_{n \times n}$ é a matriz de alocação, com $x_{ij} = 1$ se o vértice *i* é alocado ao vértice *j* e $x_{ij} = 0$, caso contrário; $x_{ii} = 1$ se o vértice *i* é uma mediana e $x_{ii} = 0$, caso contrário; *p* é o número de medianas (instalações) a serem localizadas; *n* é o número de vértices na rede e $N = \{1, ..., n\}$.

A função objetivo (1) minimiza a soma das distâncias ponderadas (d_{ij}), onde d_{ij} é o produto da distância entre os vértices v_i e v_j ($d(v_i, v_j)$) pelo peso w_j, sendo o peso w_j a demanda de cada vértice v_j, assim $d_{ij} = w_j \cdot d(v_i, v_j)$.

As restrições (2) e (4) definem que cada vértice *j* é alocado a somente um vértice *i*, que deve ser uma mediana. A restrição (3) determina o número *p* de medianas a ser localizado e as restrições (5) correspondem às condições de integralidade.

Outros métodos para solucionar o PPM têm sido apresentados na literatura, podendo-se citar o estudo de Eiben *et al*. (2012), que se utilizou dos resultados da teoria espectral na construção de um algoritmo híbrido baseado no método de Teitz & Bart e o trabalho de Gajardo *et al*. (2010), que apresenta uma heurística que reduz o número de variáveis do problema, baseando-se na determinação de uma distância máxima que divide a região do problema em sub-regiões, facilitando a abordagem posterior do método exato de programação linear. O estudo do PPM tem grande importância científica e contribui diretamente em problemas da vida em sociedade. Como exemplo, pode-se citar as aplicações na determinação da localização de pontos de coleta e transmissão de dados do processo eleitoral brasileiro [Correia *et al*., 2010] e instalações de postos de saúde pública para o atendimento de epidemias de dengue na cidade de Salvador – BA [Junior e Santos, 2010].

2.1.3 Problema de Máxima Cobertura

O Problema de Máxima Cobertura (PMC) tem como objetivo localizar um limitado número de instalações para cobrir o máximo número de pontos de demanda, mas não necessariamente todos. Tal problema ocorre tipicamente na localização de radares e de instalações de telecomunicações, como antenas, torres de transmissão, etc. Uma releitura do problema de Máxima Cobertura, conhecido com Problema de Cobertura de Conjuntos, é o caso em que se deseja atender toda a demanda e minimizar o número de instalações necessárias. Deve-se ressaltar que atender a demanda significa que o afastamento máximo, ou raio de cobertura *S,* entre os pontos de demanda e de oferta seja respeitado.

O PMC foi introduzido por Church e ReVelle (1974). A formulação matemática do PMC é:

$$Z = \max \sum_{i \in I} a_i \, y_i \, . \tag{6}$$

sujeito a:

$$\sum_{j \in N_i} x_j \geq y_i, \forall\, i \in I. \tag{7}$$

$$\sum_{j \in J} x_j = p. \tag{8}$$

$$x_j \in \{0,1\}, \forall\ j \in J. \tag{9}$$

$$y_i \in \{0,1\}, \forall\, i \in I. \tag{10}$$

Onde: $N_i = \{j \in J / d_{ij} \leq S\}$, $\forall\, i \in I$; x_j = 1, se o vértice $j \in J$ for selecionado para se tornar uma instalação e x_j = 0, caso contrário; y_i = 1, se o vértice $i \in I$ é atendido por alguma instalação e y_i = 0, caso contrário; a_i indica a demanda no vértice $i \in I$ e p é o número de instalações a serem ativadas.

A função objetivo (6) maximiza a demanda coberta; a restrição (7) garante que um cliente será atendido, se existe pelo menos uma instalação localizada dentro da distância de cobertura. A restrição (8) limita a exatamente p o número de instalações localizadas e as restrições (9) e (10) definem que as variáveis de decisão são do tipo binário.

Como exemplos de aplicações podem-se citar: Siliprande e Cortes (2008) que propuseram um modelo de Programação Linear Multi-objetivo, modelado como um Problema de Localização de Máxima Cobertura, para o Problema de Localização de antenas para Internet à rádio no município de Itaperuna-RJ. Andrade e Cunha (2014) abordaram um problema de planejamento de sistemas de atendimento emergencial, tendo como estudo de caso o município de São Paulo-SP, considerando SAMU-SP, suas bases e viaturas. Nesse estudo os autores analisam, entre outros resultados, a situação atual apresentada e indicam a quantidade de bases necessárias para se reduzir o tempo de atendimento em até 11 minutos.

2.2 MÉTODOS DE SOLUÇÃO PARA PROBLEMAS DE LOCALIZAÇÃO

Esta seção aborda os métodos mais citados na literatura para a determinação de soluções de problemas de Localização de Instalações. Estes

métodos são apresentados em quatro classes: Exatos, Heurísticos, Meta-heurísticos e Híbridos. Aos métodos híbridos foi dada uma atenção especial, pois destes, foi extraído o principal método abordado neste trabalho.

2.2.1 Métodos Exatos

O espaço de busca dos PPM e PMC, é um conjunto discreto e finito e em função disso, são classificados como problemas de otimização combinatória. De maneira geral, os problemas de otimização combinatória são fáceis de descrever, mas difíceis de resolver, e a dificuldade deve-se ao objetivo de encontrar uma melhor solução factível ou até mesmo a solução ótima, que satisfaça todas as restrições impostas no problema abordado.

Para Stefanello (2011), uma característica importante dos métodos exatos é a garantia da obtenção da solução ótima do problema. No entanto, cita-se que costumam ser eficientes apenas em instâncias de pequeno porte, pois o tempo para se encontrar a solução aumenta dramaticamente, conforme se aumenta o tamanho da instância, o que restringe o uso prático dos mesmos. Isto implica que nem sempre é possível encontrar a solução ótima através da implementação e execução dos algoritmos de solução desses métodos, com um número aceitável de operações. Este número de operações independe do *hardware* e da linguagem de programação, e é denominado de Complexidade do Algoritmo. De acordo com Garey e Johnson (1979) as principais classes de complexidade relativa à performance dos algoritmos com respeito a complexidade do tempo são:

i. Classe *P* (*Polynomial*): A Classe *P* é definida como um conjunto de todos os problemas de decisão resolvíveis por um algoritmo determinístico em tempo polinomial.

ii. Classe *NP* (*Non-deterministic polynomial*): A classe *NP* é definida como o conjunto de todos os problemas de decisão resolvíveis por algoritmo não determinístico em tempo polinomial. Uma definição alternativa para a classe NP é como a classe de problemas de decisão para os quais a verificação de que uma solução estimada para uma dada entrada satisfaz todos os requerimentos do problema, *pode* ser certificada rapidamente, isto é, em tempo polinomial.

iii. Classe *NP*-Difícil (*NP-Hard*): A classe NP-Difícil constitui-se de todos os problemas que não possuem solução determinística polinomial conhecida e que são polinomialmente equivalentes entre si. Problemas da classe NP-Difícil não necessariamente possuem uma forma eficiente de verificar suas soluções, ou seja, ao contrário dos problemas da classe *NP*, não necessariamente existem métodos polinomiais de verificar se uma resposta para um problema de decisão dessa classe é correta.

iv. Classe *NP*-Completo (*NP-Complete*): o teorema Cook-Levin, do início da década 1970, comprova que há um grupo em *NP* que se houver um algoritmo polinomial capaz de resolver qualquer um desses problemas, então todos os problemas da classe *NP* seriam solúveis em tempo polinomial. Esses problemas pertencem à classe *NP*-Completo.

Para a utilização dos métodos exatos, além da enumeração exaustiva, utilizam-se, por exemplo, técnicas de busca em árvore (*Branch-and-Bound* ou *Tree Search*), relaxação da programação inteira, métodos duais e métodos baseados em relaxação lagrangeana.

2.2.2 Métodos Heurísticos

Quando não é possível encontrar uma solução ótima para o sistema, muitas vezes em função da sua complexidade, utilizam-se os métodos heurísticos, que são capazes de fornecer uma resposta próxima ao resultado ótimo. A qualidade da solução não pode ser medida, uma vez que não se tem conhecimento da solução ótima, em problemas envolvendo instâncias de dimensões médias e grandes. No entanto, diversos estudiosos aplicam testes envolvendo alguns métodos heurísticos e meta-heurísticos existentes, como Horner (2009) e Galvão (2004), no intuito de avaliar a qualidade da solução apresentada por estes, para problemas aleatórios, e os resultados apontam soluções próximas às soluções ótimas, nos diversos casos.

Os métodos aproximativos, que fornecem soluções dentro de um limite de qualidade absoluto ou assintótico, podem se enquadrar nesta categoria. Além disso, é comum, conforme Osman e Laporte (1996), os algoritmos aproximativos serem tratados como algoritmos heurísticos.

O estudo da aplicação de métodos heurísticos para resolver Problemas de *P*-Medianas iniciou-se com Kuehn e Hamburger (1963), que propuseram a heurística gulosa, também conhecida como método da adição de vértices que consiste em adicionar, recursivamente, medianas à solução. Iniciando com uma mediana e adicionando outras, sequencialmente até que *p* medianas tenham sido incluídas. Em 1964, Maranzana utilizou uma técnica que ficou conhecida como Método da Partição de Vértices, por buscar sucessivos vértices únicos de *m* subconjuntos destinos, cada um associado a uma origem. Estes conjuntos vão sendo modificados e o processo é repetido até que se tenha uma solução satisfatória. Ainda na década de 60, Teitz e Bart (1968), propuseram um dos primeiros métodos heurísticos, com bons resultados para a época, para se encontrar a mediana de um grafo ponderado. O método procura a solução para o problema através da troca de vértices, a partir de uma solução inicial.

Segundo Sauer (2007), as heurísticas para Problemas Combinatórios abrangem os seguintes tipos: métodos construtivos, métodos de enumeração limitada e métodos de melhoria. Romero e Montovani (2004) afirmam que os métodos construtivos baseiam-se na construção de uma solução viável para o problema partindo de uma solução trivial, e sobre a qual são aplicadas diferentes técnicas, vindo melhorar a qualidade da solução a ser obtida. No método, são considerados os conjuntos *C*, de objetos alocados e L_o, de localizações ocupadas, ambos inicialmente vazios. Em tais métodos a construção de uma permutação π é feita de forma heurística, escolhendo-se a cada passo, a próxima alocação (*i*, *j*), tal que $i \notin C$, $j \notin L_o$, e fazendo-se $\pi(i) = j$. Para um problema de ordem *n*, o processo é repetido até completar uma permutação na ordem do problema [Rosenkrantz *et al.* (1977)].

Os métodos de enumeração, ou buscas exaustivas, são simples. Estipula-se um espaço finito de busca e o algoritmo verifica todas as combinações possíveis de soluções. Observa-se que somente podem garantir que a solução encontrada é ótima se chegarem ao final do processo enumerativo. No entanto, é muito comum que uma boa solução, ou até mesmo uma solução ótima, seja encontrada no início da enumeração. Observa-se que, quanto melhores são as informações utilizadas para guiar a enumeração, maiores serão as chances de se encontrar prematuramente soluções de boa qualidade. No entanto, garantir o ótimo para a solução encontrada, em geral, é muito demorado. Para limitar esta enumeração, são

definidas condições de parada, tais como: um número máximo de iterações realizadas pelo algoritmo; finalizar o algoritmo se após um número pré-determinado de passos não ocorrer nenhuma melhoria da função custo; tempo máximo de execução, etc. Torna-se bastante claro que qualquer um destes critérios de parada pode ocasionar a eliminação da solução ótima, o que nos leva a tomar certos cuidados ao utilizar métodos de enumeração limitada. [Burkard & Bonniger, 1983].

Os métodos de melhoria compreendem os algoritmos de busca local e iniciam com uma solução viável, para na sequência tentar melhorá-la, procurando outras soluções em sua vizinhança. O processo é repetido até que nenhuma melhoria possa ser encontrada. Os métodos nesta categoria são comumente utilizados pelas meta-heurísticas.

2.2.3 Meta-heurísticas

Na busca por um método mais geral, chegou-se ao desenvolvimento de métodos chamados meta-heurísticos.

Segundo Hillier e Lieberman (2010), “meta-heurística é um método de resolução geral, que fornece tanto uma estrutura, quanto diretrizes de estratégias gerais para desenvolver um método heurístico específico que se ajuste a um tipo de problema particular”. Afirmaram ainda que as três meta-heurísticas mais comumente utilizadas são: Busca Tabu, *Simulated Annealing* e Algoritmos Genéticos.

Como os Problemas de *P*-Medianas e Máxima Cobertura são problemas da classe *NP-Hard*, a aplicação de meta-heurísticas é indicada.

Pesquisadores que se utilizaram do algoritmo Busca Tabu, confirmam sua eficiência em aplicações diversas.

> Diversas aplicações em uma grande variedade de áreas têm disponibilizado estudos computacionais documentando sucessos da Busca Tabu em entender a fronteira de problemas que podem ser ligados de forma eficiente, levando a soluções cuja qualidade se sobrepõe significantemente a aquelas obtidas por métodos aplicados previamente [Gomes, 2009].

Segue a apresentação de alguns trabalhos que utilizaram o algoritmo Busca Tabu, presentes na literatura.

O algoritmo BT foi utilizado, após se obter uma solução inicial gerada por um procedimento construtivo, por Subramanian *et al.* (2011) a um problema de alocação de aulas a salas em uma instituição educacional onde demonstrou ser bastante eficiente, tendo gerado soluções de alta qualidade quando comparado com a solução obtida manualmente.

Gendreau *et al.* (1994) apresentou uma nova pesquisa heurística baseada no algoritmo Busca Tabu, denominado pelos autores de TABUROUTE, para o problema de roteamento de veículos com restrições de capacidade e duração da rota. O movimento básico é feito através de permutação de vértices, utilizando um procedimento que consiste em inserir um vértice entre dois dos seus p vizinhos mais próximos na rota enquanto realiza uma reotimização local da rota. Outra característica é a busca estar oscilando entre soluções viáveis e inviáveis.

El Rhazi e Pierre (2009) propõem um novo método de agrupamento centralizado de um mecanismo de coleta de dados em redes de sensores sem fio (*wireless*), que é baseado em mapas de energia de rede e de qualidade de serviço requisitada pelos usuários. O problema de agrupamento é modelado como um particionamento hipergrafo e sua resolução é baseada em uma heurística de Busca Tabu. Segundo os autores, comparado a outros métodos, os resultados mostram que a abordagem baseada em Busca Tabu, retorna soluções de alta qualidade em termos de custo de *cluster* e tempo de execução.

As abordagens de melhorias para as meta-heurísticas não são necessariamente, técnicas de solução autônomas, ou seja, que operam independentemente. Muitas vezes, uma série de métodos pode ser combinada de modo a obter uma solução, formando os métodos híbridos.

2.2.4 Métodos Híbridos

A evolução dos computadores nos últimos anos, caracterizada pelo aumento sistemático da capacidade de processamento e avanço nas técnicas de paralelização, tem permitido a avaliação e solução de problemas computacionais num tempo inferior, se comparado a computadores utilizados há poucos anos. Acrescenta-se ainda, a melhora no desempenho dos algoritmos otimizadores para programação inteira, cuja eficiência é evidente diante das pesquisas apresentadas

na literatura. Segundo Nievergelt (2000), é aceito que não existe algoritmo capaz de resolver inteiramente uma classe de problemas *NP-hard*, mas ignora-se o fato de que muitas instâncias dos problemas desta classe, inclusive as de interesse prático, podem ser resolvidas eficientemente.

Com o objetivo de melhorar a eficiência das meta-heurísticas, diversas abordagens apresentadas na literatura, procuram adicionar buscadores locais ou fazer combinações com outras técnicas de resolução. Estas abordagens são, de forma geral, denominadas de hibridização. Torna-se então propício a hibridização de métodos, pois explora as melhores características de cada um destes métodos, principalmente considerando que, geralmente, as meta-heurísticas são ágeis e especializadas enquanto os métodos exatos são demorados e, em muitos casos, genéricos.

Em Jourdan *et al*. (2009), os autores apresentam um levantamento e classificação de várias abordagens encontradas na literatura, que fazem uso de hibridização, mostrando o número crescente de publicações relacionadas, o que comprova o interesse que este assunto vem despertando na comunidade científica.

Os autores ainda apresentam uma extensão do trabalho realizado em Talbi (2002), no qual, além de fazer uma breve revisão da literatura, também mostram como os métodos podem ser combinados e como pode ser a cooperação entre os mesmos. Segundo a classificação, a taxonomia é dividida em três aspectos gerais: projeto de métodos cooperativos (*cooperation method design*), projeto de modelos de abordagens (*approach design*) e questões de implementação (*implementation issues*).

Yen *et al*. (1998) classificaram em quatro categorias os Algoritmos Evolutivos Híbridos, a saber:

i. Híbridos *pipelining*: as meta-heurísticas e/ou técnicas de otimização são aplicadas sequencialmente; uma gera dados (ou seja, pontos no espaço de busca) usados pela(s) outra(s). Normalmente, os híbridos *pipelining* usam um algoritmo de busca para podar o espaço de busca inicial, de tal forma que o segundo algoritmo convirja mais rapidamente ou com mais precisão.

ii. Híbridos assíncronos: utiliza uma população compartilhada com várias técnicas de otimização para prosseguir e cooperar de forma assíncrona. Todas as

técnicas que compõem a hibridização trabalham de forma cooperativa, podendo modificar as soluções das populações em comum.

iii. Híbridos hierárquicos: usam técnicas de otimização em diversos níveis, diferentes de um problema de otimização.

iv. Algoritmo com operadores adicionais: um algoritmo pode, às vezes, ser melhorado, incluindo-se operadores com movimentos de busca.

Alguns trabalhos que envolvem hibridização com Algoritmos Evolutivos são comentados a seguir.

Almeida, Pacheco e Vellasco (2003) propuseram um algoritmo híbrido usando Algoritmo Cultural e Algoritmo Genético, como apoio para encontrar melhores alternativas para o desenvolvimento de um campo de petróleo.

Em Fernandes e Lourenço (2008), as autoras apresentam um mapeamento dos trabalhos que usam procedimentos híbridos baseados na combinação de meta-heurísticas e métodos exatos levando em consideração as classificações propostas por Dumitrescu e Stützle (2003) e Puchinger e Raidl (2005).

Nos trabalhos Mautor e Michelon (1997 e 2001) os autores apresentam o método MIMAUSA. Este método foi usado no problema quadrático de alocação em que a busca local pela vizinhança é definida eliminando algumas variáveis, e resolvendo o subproblema de forma exata.

Carrano *et al*. (2005) apresentaram um algoritmo híbrido que combina um algoritmo quase-Newton e um Algoritmo Genético para a expansão ótima de sistemas de distribuição de energia. O procedimento quase-Newton atua na busca das coordenadas de localização ótima de subestação e o Algoritmo Genético atua no projeto de topologias de rede ótima. Ambos os buscadores são realizados em conjunto na metodologia proposta. O algoritmo foi aplicado a um problema real de expansão de sistema de distribuição de energia.

Duvvuru e Swarup (2011) propuseram um algoritmo híbrido agrupando o método de ponto interior (MPI) e a Evolução Diferencial para resolver o problema de despacho econômico de carga, com efeitos de ponto de válvula. O algoritmo envolve duas etapas. A primeira etapa emprega MPI para minimizar a função de custo, sem considerar o efeito de ponto de válvula. A segunda etapa considera efeito de ponto de válvula e minimiza a função custo utilizando Evolução Diferencial.

Rosário (2011) abordou um problema de Programação de Pessoal com aplicação em uma empresa de *Call Center*. O problema foi dividido em duas etapas: problema de Turnos de Trabalho e problema de Designação dos Turnos aos Atendentes. A primeira etapa foi resolvida utilizando-se de Algoritmos Genéticos e um algoritmo de Evolução Diferencial modificado para espaços discretos. Para a resolução da segunda etapa, foi desenvolvido um Algoritmo Evolutivo que integra outros Algoritmos Evolutivos, denominado Algoritmo Evolutivo Adaptativo.

Nas próximas seções apresentam-se: o algoritmo Teitz & Bart, utilizado para comparações de resultados com o algoritmo proposto, descritos no Capítulo 4 e os algoritmos Busca Tabu e Evolução Diferencial, utilizados na construção do algoritmo híbrido proposto neste trabalho.

2.3 TEITZ & BART

O método heurístico baseado na substituição de vértices, desenvolvido por Teitz e Bart (1968), destaca-se entre os diversos métodos heurísticos e metaheurísticos utilizados para a resolução do problema de *P*-Medianas. Testes computacionais, comparando o algoritmo Teitz & Bart e Busca Tabu, realizados por Hörner (2009), concluíram que o método de Teitz & Bart apresenta melhor qualidade de solução, apesar do tempo computacional não ter sido considerado satisfatório.

Considerando-se todos os vértices de um grafo dado como medianas potenciais, o algoritmo Teitz & Bart para o problema das *P*-Medianas pode ser explicado como segue: seja $G = (V,A)$ um grafo não direcionado onde V são os vértices e A as arestas. Seja v_i um vértice qualquer pertencente a V. Chama-se número de transmissão à soma das menores distâncias existentes entre o vértice v_i e todos os outros vértices do grafo.

Considerando n o número total de vértices do grafo, o número de transmissão é dado por:

$$\sigma(v_i) = \sum_{j=1}^{n} w_j \, d(v_i, v_j), \quad v_i, v_j \in V \tag{11}$$

Onde, $d(v_i, v_j)$ é a menor distância entre v_i e v_j e w_j é um peso associado ao vértice v_j.

Assim, v_m é uma mediana se, entre todos os vértices do grafo, v_m é aquele que produz a menor soma total das distâncias, desde si próprio até cada um dos outros vértices do grafo. Ou seja:

$$\sigma\ (v_m) = \text{mínimo}\ [\sigma(v_i)],\ \forall\ v_i \in\ V \tag{12}$$

Para o problema de encontrar *p*-medianas (p > 1), seja $S \subset V$ e $|S| = p$, calculam-se:

$$d(S,v_j) = \text{mínimo}\ [d(v_i,v_j)],\ \forall\ v_i \in\ S\ ,\ v_j \in\ V \tag{13}$$

e

$$\sigma(S) = \sum_{j=1}^{n} w_j\ d(S, v_j),\ \forall\ v_j \in\ V \tag{14}$$

Um conjunto *S* de *p* vértices é a solução ótima para o problema das *P*-Medianas se entre todos os outros conjuntos de *p* vértices do grafo, *S* é aquele que produz a menor distância total desde si próprio até os outros vértices do grafo. Portanto, deve-se ter:

$$\sigma(S_{\_solução\_ótima}) = \text{mínimo}\ [\sigma(S)],\ \forall\ S \subset V \tag{15}$$

2.3.1 Descrição do algoritmo Teitz & Bart

São descritos a seguir os procedimentos básicos executados pelo algoritmo das *P*-Medianas de Teitz & Bart.

Passo 1 :

Selecione aleatoriamente um conjunto $S \subset V$, com *p* elementos, criado para possibilitar uma aproximação inicial para o problema das *P*-Medianas.

Passo 2:

Rotule todos os vértices v_i, pertencentes a $\{V - S\}$ como “não analisados”.

Passo 3:

Enquanto existirem vértices não analisados em $\{V - S\}$, onde V é o conjunto de vértices do grafo. Faça:

- Selecione um vértice “não analisado” $v_i \in \{V - S\}$ e calcule a redução $\Delta i\, j$ do número de transmissão para todo $v_j \in S$, *ou seja:*

$$\Delta_{i\,j} = \sigma(S) - \sigma(S \cup \{v_i\} - \{v_j\}) \qquad (16)$$

- Faça $\Delta_{i\,j0} = max\,[\Delta_{i\,j}]$;
- Se $\Delta_{i\,j0} > 0$, faça $S \leftarrow S \cup \{v_i\} - \{v_{j0}\}$ e rotule v_{j0} como “analisado”;
- Se $\Delta_{i\,j0} \leq 0$, rotule v_i como “analisado”;

Passo 4:

Se durante a execução do passo 3 houver alguma modificação no conjunto S, volte ao Passo 2. Caso contrário, pare e apresente o conjunto S como uma aproximação para solução do problema de P-Medianas.
Fim.

Horner (2009) apresenta, no Apêndice A, a resolução de um exemplo didático, pelo método de Teitz & Bart.

2.4 BUSCA TABU

A meta-heurística Busca Tabu (BT) foi inicialmente desenvolvida por Glover (1986), como uma proposta de solução para problemas de programação inteira. A partir de então, o autor publicou uma série de trabalhos contendo diversas aplicações da mesma. A experiência tem mostrado a eficiência da Busca Tabu na resolução de vários problemas de diferentes naturezas (Glover e Laguna, 1997) e, atualmente, pode-se afirmar que se trata de uma técnica consolidada.

O algoritmo Busca Tabu é um procedimento adaptativo, de busca local, dotado de uma estrutura de memória, que aceita movimentos de piora, quando não há expectativa de melhora, para escapar de ótimos locais [Higgins, 2001; Souza, 2000]. Sendo um procedimento de busca local, é baseado na noção de vizinhança. A cada iteração, a solução atual *s* muda para outra que seja sua vizinha no espaço de busca, isto é, para uma solução *s'* que difere de *s* por uma modificação.

Partindo de uma solução inicial s_0 (FIGURA 1), o algoritmo Busca Tabu explora, a cada iteração, um subconjunto *S* de vértices vizinhos da solução corrente *s* (FIGURA 2).

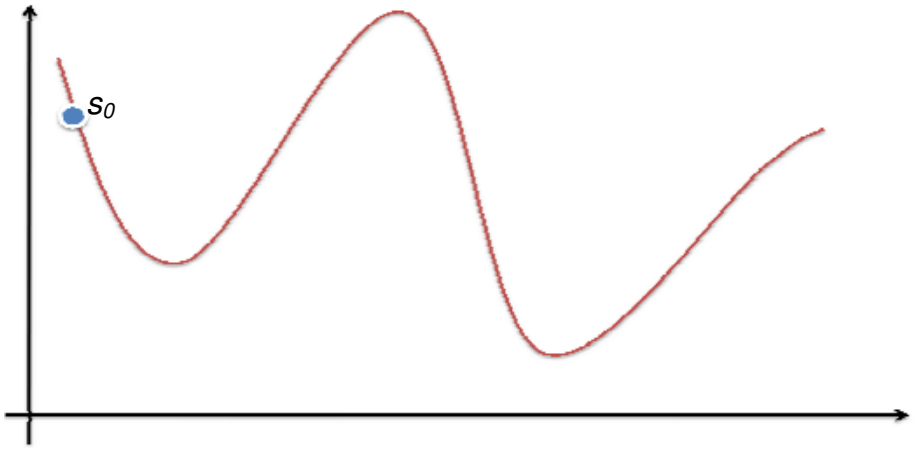

FIGURA 1 - SOLUÇÃO INICIAL (s_0)

FIGURA 2 - ÓTIMO LOCAL

O algoritmo não aceita movimentos que levem a soluções já visitadas, pois os mesmos permanecem armazenadas em uma Lista Tabu (FIGURA 3).

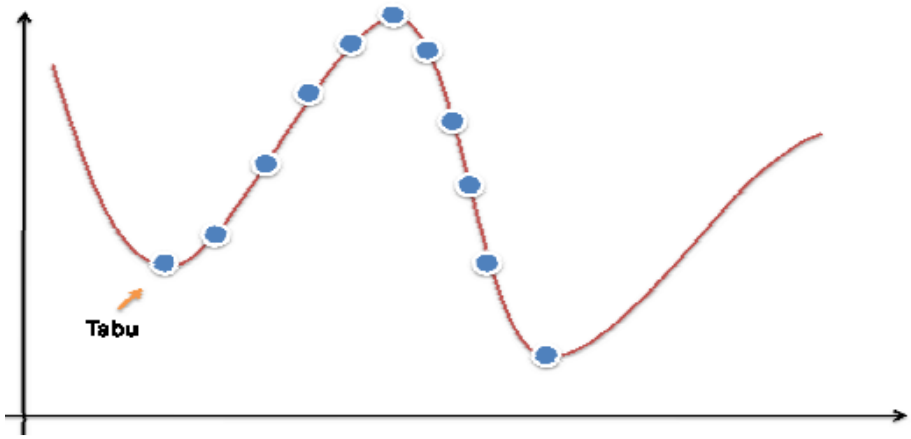

FIGURA 3 - ELEMENTOS TABU

A lista *permanece* na memória, guardando soluções já visitadas (Tabu), durante um determinado espaço de tempo ou certo número de iterações (prazo tabu). Como resultado final é esperado que se encontre um ótimo global ou a solução mais próxima deste (FIGURA 4).

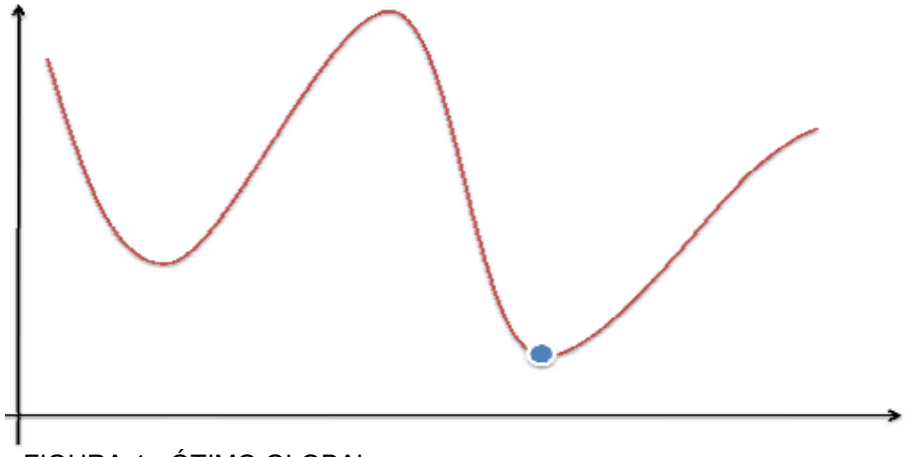

FIGURA 4 - ÓTIMO GLOBAL

A Busca Tabu utiliza estruturas de memória flexíveis para armazenar o conhecimento referente aos espaços que já foram percorridos, devido ao fato de utilizar intensivamente técnicas de memória adaptativa. A técnica consegue proporcionar soluções adequadas, isto é, soluções geralmente ótimas ou perto do ótimo global, em um tempo de processamento aceitável. Outra importante característica do método é que a solução final tem pouca ou nenhuma dependência da escolha feita para a solução inicial. Isso graças aos mecanismos implementados pelo método que escapam de ótimos locais. [Gomes, 2009].

2.4.1 Implementação do algoritmo Busca Tabu

De forma simplificada, pode-se ver o método como uma técnica iterativa que explora um conjunto de soluções *S* de um problema, repetidas vezes, fazendo movimentos de uma solução *s* para outra solução *s'* localizada na vizinhança *V*(*s*).

Partindo de uma solução inicial s_0, um algoritmo Busca Tabu explora, a cada iteração, um subconjunto *S* de vértices vizinhos da solução corrente *s*. O membro *s'* de *S* com melhor valor nessa região segundo a função aptidão *f*, torna-se a nova solução corrente mesmo que *s'* seja pior que *s,* isto é, que $f(s') > f(s)$ para um problema de minimização.

O uso da memória é uma característica essencial da Busca Tabu. Enquanto a maioria dos processos de busca, guarda essencialmente o valor de $f(s^*)$ da melhor solução s^* obtida até o momento, a Busca Tabu arquiva as informações em uma espécie de itinerário das últimas soluções visitadas. Tais informações são utilizadas para conduzir o movimento de uma solução para outra a ser escolhida no espaço de busca. A função da memória consiste em restringir a escolha de algum vértice, proibindo movimentos para algumas soluções vizinhas (Hertz *et al.*, 1995).

A proibição desses movimentos tem a intenção de impedir o retorno a uma solução já visitada anteriormente. O não veto de determinados movimentos pode fazer com que o algoritmo torne-se cíclico. Um artifício criado com o intuito de não "autorizar" a ocorrência desses movimentos é a Lista Tabu, que consiste em uma lista contendo as soluções visitadas durante as últimas $|K|$ iterações sequenciadas na forma FIFO (*First In First Out*). O algoritmo apresenta ainda uma função de aspiração que indica um critério para retirar o *status* tabu em certas circunstâncias que podem ser, por exemplo, aspiração por objetivos, onde se aceita o movimento tabu se ele melhorar o valor da função objetivo corrente.

Os principais parâmetros de controle do método são o tamanho da Lista Tabu, a cardinalidade do subconjunto *S* das soluções vizinhas testadas a cada iteração e o número máximo de iterações sem melhora na função objetivo.

A seguir é mostrado o pseudo-código do algoritmo Busca Tabu no QUADRO 1.

QUADRO 1 – PROCEDIMENTO BUSCA TABU

PROCEDIMENTO DO ALGORITMO BUSCA TABU
01. Seja s_0 solução inicial; 02. $s' \leftarrow s$; {Melhor solução obtida até então} 03. *Iter* $\leftarrow 0$; {Contador do número de iterações} 04. *MelhorIter* $\leftarrow 0$; {Iteração mais recente que forneceu s'} 05. Seja *BTmáx* o número máximo de iterações sem melhora em s'; 06. $T \leftarrow \varnothing$; {Lista Tabu} 07. Inicialize a função de aspiração *A*; 08. Enquanto (*Iter* – *MelhorIter* ≤ *BTmáx*) faça 09. *Iter* $\leftarrow$ *Iter* + 1; 10. Seja $s' \leftarrow s \oplus m$ o melhor elemento de $S \subseteq N(s)$ tal que o movimento *m* não seja tabu ($m \notin T$) ou s' atenda a condição de aspiração ($f(s')<A(f(s))$; 11. Atualize a Lista Tabu *T*; 12. $s \leftarrow s'$ 13. Se $f(s) < f(s')$ então 14. $s' \leftarrow s$ 15. *MelhorIter* $\leftarrow$ *Iter;* 16. Fim – se 17. Atualize a função de aspiração *A*; 18. Fim – enquanto; 19. Retorne s^*; Fim *BT*;

FONTE: GOMES (2009)

Na próxima seção serão abordados os princípios básicos de um procedimento meta-heurístico denominado Evolução Diferencial, que faz parte dos Algoritmos Evolutivos.

2.5 ALGORITMOS DE EVOLUÇÃO

Problemas de otimização estão presentes em muitas áreas do conhecimento, principalmente em Ciências e Engenharia. Sem dúvida, muitos pesquisadores precisam de um algoritmo de otimização robusto para resolver os problemas que são fundamentais para seu trabalho diário. Além de fácil de usar, um algoritmo de otimização global também deve ser suficientemente robusto para convergir de forma confiável para o verdadeiro ótimo. Além disso, o tempo computacional na procura de uma solução não deve ser excessivo. Assim, um método de otimização global, verdadeiramente útil, deve ser simples de implementar, fácil de usar, confiável e rápido [Price *et al.*, 2005].

Os algoritmos iterativos (busca e otimização) desenvolvidos com inspiração no processo biológico de evolução são denominados Algoritmos Evolutivos (AE's). Os Algoritmos Evolutivos se tornaram uma opção para a otimização de problemas complexos demais para serem resolvidos por técnicas determinísticas, como os métodos de programação linear e gradiente [Sbalzariniy *et al.*, 2000]. Os AE's são métodos probabilísticos que imitam o processo de evolução natural e desde os anos 80 têm sido utilizados para a resolução de problemas difíceis de otimização, como relatado em [Bäck *et al.*, 1997]. A ideia básica por trás de todos estes métodos é essencialmente a mesma: dada uma população de indivíduos, a pressão ambiental causa seleção natural o que leva à sobrevivência do mais adaptado. Ao longo do tempo esta seleção melhora a aptidão dos indivíduos da população. Adaptando esta ideia para o processo de otimização, temos uma população formada por um conjunto de soluções candidatas ao problema. Estas soluções podem sofrer algum tipo de variação e, ao longo do tempo, têm sua adaptabilidade medida através da função objetivo. Soluções mais adaptadas, ou seja, soluções que possuem um valor melhor de função objetivo são selecionadas e repetem o processo. As soluções ruins são descartadas.

A principal vantagem no uso de AE's está no ganho de flexibilidade e adaptabilidade em relação ao problema em estudo, em combinação com desempenho robusto e com suas características de busca global [Bäck *et al.*, 1997].

O esquema básico de um algoritmo evolutivo pode ser visto na FIGURA 5 a seguir:

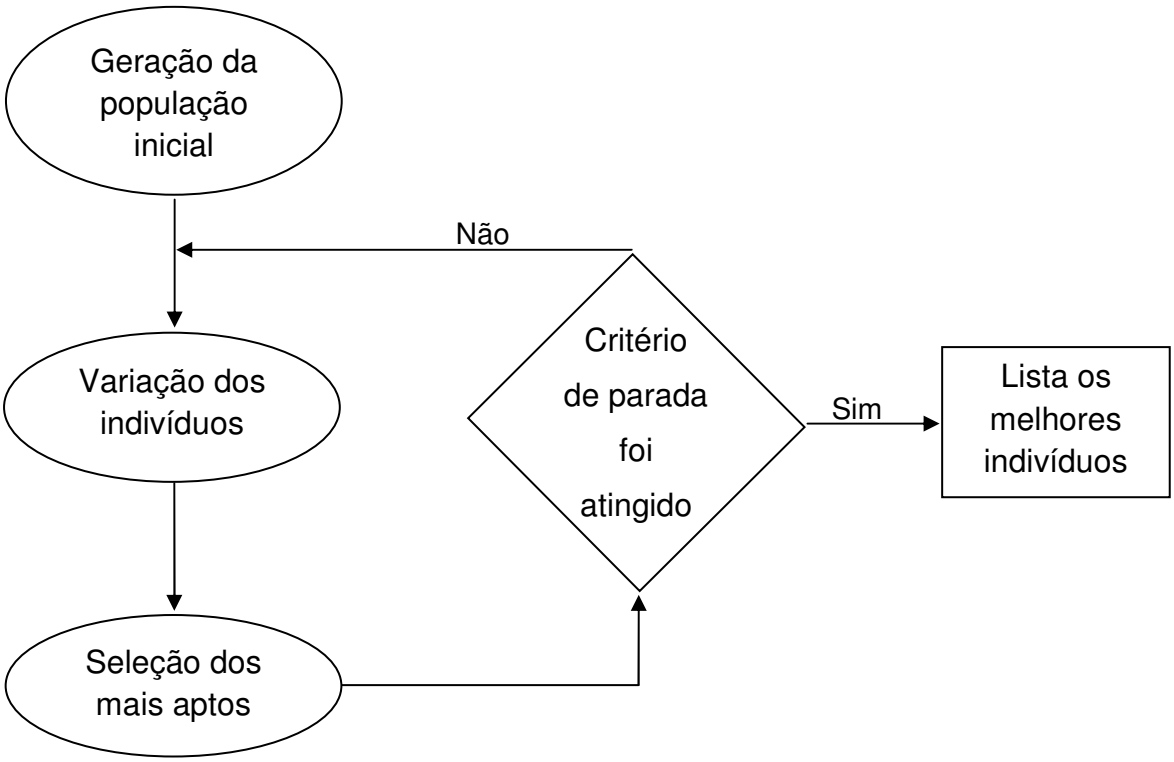

FIGURA 5 – ALGORITMO EVOLUTIVO BÁSICO
FONTE: SILVA (2010)

Computação Evolutiva é o nome usado para descrever o campo de pesquisa que inclui todos os Algoritmos Evolutivos [Castro, 2006].

Nas últimas décadas, os avanços obtidos na área dos AE's aumentaram o domínio de aplicações dessas ferramentas e melhoraram seu desempenho em diversos contextos. Entre essas ferramentas, destaca-se o algoritmo de Evolução Diferencial, que é um otimizador simples e versátil para a otimização de funções com variáveis contínuas.

2.6 EVOLUÇÃO DIFERENCIAL

Evolução Diferencial (ED), proposto por Price e Storn (1995 e 1997), originou-se por meio do algoritmo *Annealing* Genético desenvolvido por Kenneth Price, publicado em outubro de 1994 numa revista destinada a programadores, chamada *Dr. Dobb's Journal*. *Annealing* Genético é um algoritmo de otimização combinatória, baseado em população, que implementa um critério *annealing* via

limiares. Após esta publicação, Price foi contactado pelo Dr. Rainer Storn, sobre a possibilidade de utilizar *Annealing* Genético para resolver o problema polinomial de Chebyshev, pois, determinar os coeficientes dos polinômios de Chebyshev é considerado por muitos uma tarefa difícil para um otimizador de propósito geral [Price *et al*., 2005].

Price, finalmente, encontrou a solução para o problema cinco-dimensional de Chebyshev com o algoritmo *Annealing* Genético, mas a convergência foi muito lenta e os parâmetros de controle efetivos foram difíceis de determinar. Após este resultado inicial, Price começou a modificar o algoritmo *Annealing* Genético usando ponto flutuante em vez de bits de sequência de codificação e operações aritméticas em vez de lógicas. Em conjunto, estas alterações efetivamente transformaram o que tinha sido um algoritmo combinatório para otimizador numérico, que se tornou a primeira experiência com ED. Para melhor acomodar arquiteturas de máquinas paralelas, Storn sugeriu a criação de população de pais e filhos separados. Ao contrário do *Annealing* Genético, a ED não tem dificuldades em determinar os coeficientes, mesmo, do polinômio 33-dimensional de Chebyshev [Price *et al*., 2005].

ED mostrou eficácia não só sobre os polinômios de Chebyshev, mas também em muitas outras funções-teste. Em 1995, Storn e Price apresentaram "*Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*" [Price e Storn, 1995], primeiro artigo de divulgação do então promissor algoritmo Evolução Diferencial. Nesse artigo foi apresentada a ED para minimizar funções de espaço contínuo não linear e não diferenciável. Por meio de exaustivos testes experimentais, que incluiu as funções “De Jong”, foi demonstrado que o método converge mais rapidamente do que o *Simulated Annealing*, considerado eficiente para essa aplicação. O novo método, o ED, requer poucas variáveis de controle, é robusto e fácil de usar.

2.6.1 Inicialização do algoritmo Evolução Diferencial

As principais etapas de um algoritmo ED clássico estão mostradas na FIGURA 6. Para cada geração, os operadores são repetidos até que um critério de parada pré-definido seja satisfeito, como por exemplo, número máximo de gerações [Price e Storn, 1997].

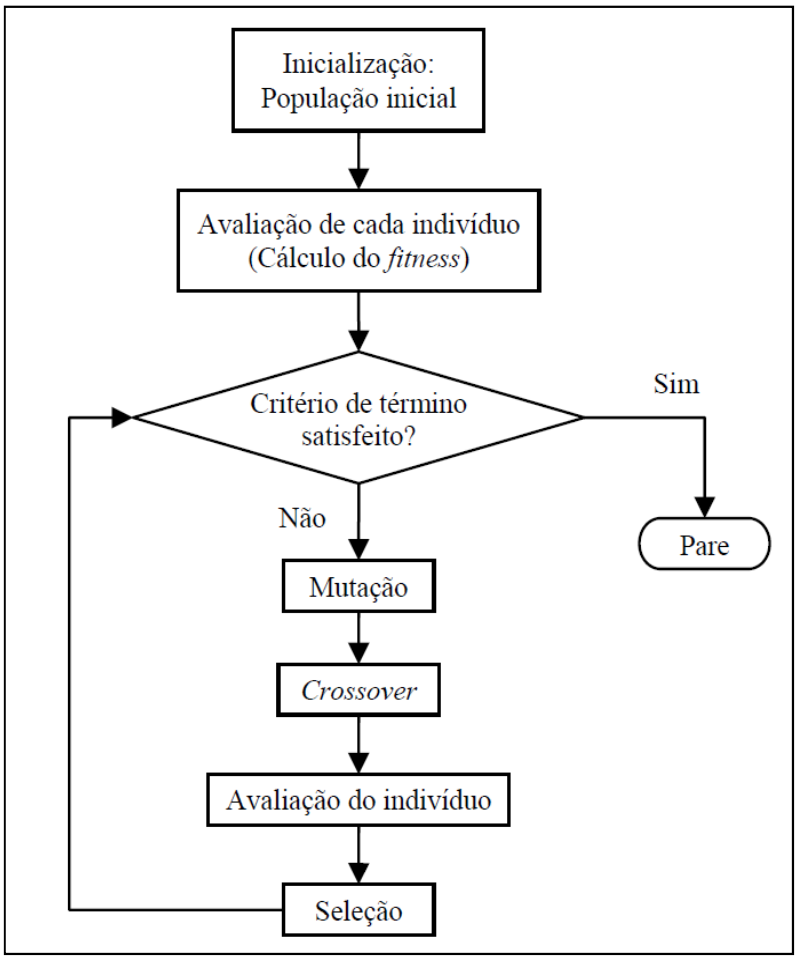

FIGURA 6 - ETAPAS DE EVOLUÇÃO DIFERENCIAL TRADICIONAL
FONTE: ROSÁRIO (2011)

Em ED, a essência da operação de mutação é a formação de um vetor diferença que faz mutação no indivíduo. A seguir apresenta-se uma breve descrição sobre cada etapa.

2.6.1.1 Avaliação dos indivíduos (Função *Fitness*)

Para avaliar um indivíduo, é necessário medir sua adaptabilidade, qualidade ou valor da função *fitness*. A função *fitness*, ou função objetivo, a ser otimizada fornece o mecanismo para a avaliação de cada indivíduo, o qual é utilizado pelo mecanismo de seleção para avaliá-los na população. A função de avaliação deve ser capaz de discriminar os indivíduos bons dos ruins, e esta diferenciação deve ser significativa, caso contrário, haverá um elevado consumo de tempo computacional.

2.6.1.2 População Inicial

Um problema de otimização, que consiste em *D* parâmetros, pode ser representado por um vetor *D*-dimensional. Como um membro da família de Algoritmos Evolutivos, ED também inicia com uma população $G = \{V_1, V_2, ..., V_i, V_{Np}\}$ de *Np* indivíduos (vetores-solução), construída geralmente de forma aleatória, dentro dos limites do espaço de busca. Quando não existe nenhum conhecimento sobre o problema, normalmente, a população é gerada por uma distribuição de probabilidade uniforme.

O tamanho da população influencia o desempenho do algoritmo. Uma população pequena pode reduzir o espaço de busca, diminuindo as possibilidades de se atingir o ótimo global. No outro extremo, as populações grandes acarretam maior consumo de tempo computacional.

2.6.1.3 Operador de mutação

Numa operação definida como mutação, o algoritmo ED gera novos vetores de parâmetros, chamados de vetores doadores, através da adição da diferença ponderada entre dois vetores a um terceiro indivíduo, conforme operação a seguir:

$$V_{doador} = V_i + F(V_j - V_k) \quad (17)$$

ou

$$V_{doador} = V_{melhor} + F(V_j - V_k) \quad (18)$$

Onde: V_{doador} é o vetor doador, F é um peso escalar aplicado ao vetor diferença, V_{melhor} é o vetor da população que apresenta o melhor *fitness* e V_i , V_j e V_k representam indivíduos aleatórios e mutuamente distintos, escolhidos da população. De acordo com a definição, o tamanho da população deve ser superior a três ($N_p \geq 3$). A mutação, como descrita, amplia o espaço de busca. Esta forma de gerar um vetor mutante é conhecida como esquema ED1 [Price e Storn, 1995].

A FIGURA 7, adaptada de Price e Storn (1995), mostra um exemplo bidimensional, ilustrando os vetores que participam da geração do vetor doador V_{doador}, a partir da equação (17) ou (18).

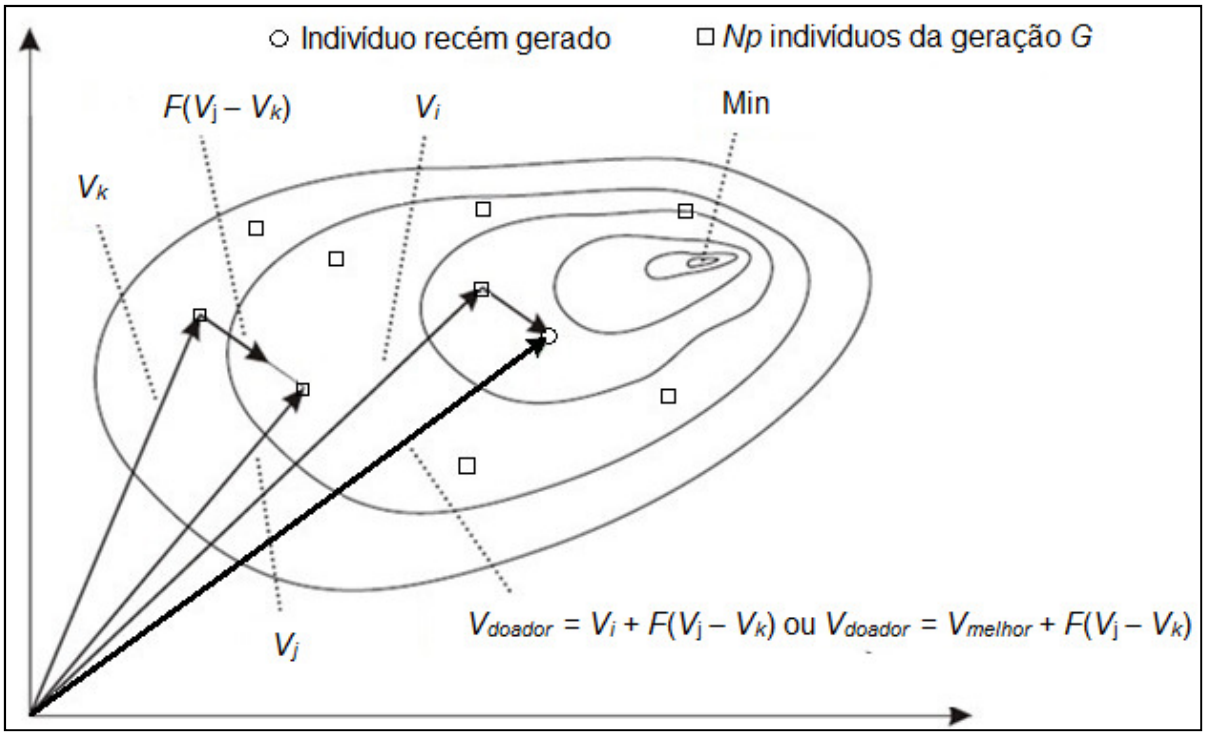

FIGURA 7 – PROCESSO DE GERAÇÃO DO VETOR MUTANTE (ED1)
FONTE: ROSÁRIO (2011)

Outro esquema frequentemente utilizado é chamado ED2 em que os vetores mutantes são produzidos por incluir o impacto do melhor vetor, V_{melhor}, da geração atual, da seguinte forma [Price e Storn, 1995, 1997]:

$$V_{doador} = V_i + F_1 (V_{melhor} - V_k) + F_2 (V_j - V_k) \quad (19)$$

Onde, i, j e $k \in \{1, 2, ..., N_p\}$ são índices mutuamente distintos escolhidos aleatoriamente e F_1 e F_2 são valores escalares que oferecem maior diversidade no processo de busca.

Caso a quantidade de indivíduos da população seja considerada elevada, pode-se melhorar a diversidade da população com o uso de duas diferenças ponderadas, a fim de perturbar um vetor existente, ou seja, cinco vetores mutuamente distintos são escolhidos aleatoriamente na população atual. O vetor diferença ponderada usa dois pares de diferenças ponderadas e é utilizado para perturbar o quinto vetor ou o melhor vetor da população atual. De acordo com [Arantes, Oliveira e Saramago, 2006]:

$$V_{doador} = V_i + F\,(V_j - V_k + V_w - V_p) \quad (20)$$

ou

$$V_{doador} = V_{melhor} + F\,(V_j - V_k + V_w - V_p) \quad (21)$$

Os índices aleatórios *i, j, k, w* e $p \in \{1, 2,..., N_p\}$ são índices inteiros mutuamente distintos escolhidos aleatoriamente e o tamanho da população, N_p, deve ser maior que 5. Este esquema é chamado de ED3.

2.6.1.4 Cruzamento (*Crossover)*

Após realizada a mutação, escolhe-se aleatoriamente outro vetor, denominado vetor alvo (V_{alvo}), cujas componentes são misturadas com as componentes do vetor V_{doador}, resultando no vetor chamado experimental (V_{exp}). Este tipo de cruzamento, denominado binomial ocorre com probabilidade de cruzamento $CR \in [0,1]$ e segue a seguinte regra:

$$e_i = \begin{cases} a_i, & \text{se } \theta_i > CR \\ d_i, & \text{se } \theta_i \leq CR \end{cases} \text{ onde } i = 1,2, \ldots, n. \quad (22)$$

Sendo θ_i números aleatórios pertencentes ao intervalo [0,1], e_i, d_i e a_i são respectivamente componentes dos vetores V_{exp}, V_{doador} e V_{alvo}. *CR* é a probabilidade de ocorrer cruzamento em ED e representa a probabilidade do vetor V_{exp} herdar os valores d_i das componentes do vetor V_{doador}. *CR* é fornecida pelo usuário e deve estar no intervalo [0, 1[. Quando *CR* for igual a 1, todas as componentes do vetor V_{exp} derivarão do vetor V_{doador}. Por outro lado, se *CR* for zero, todas as componentes do vetor V_{exp} derivarão do vetor alvo V_{alvo}.

A FIGURA 8 apresenta um exemplo de cruzamento binomial.

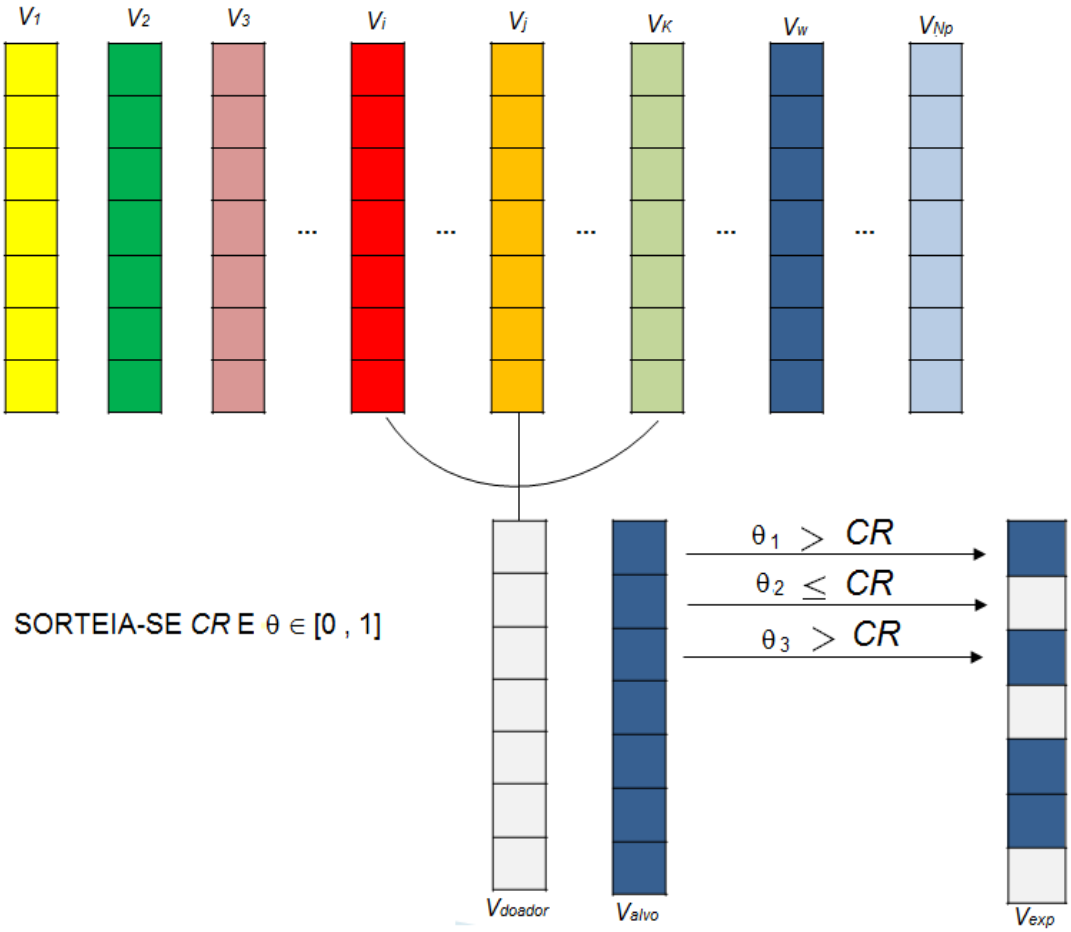

FIGURA 8 – EXEMPLO DO PROCESSO DE CRUZAMENTO BINOMIAL
FONTE: AUTOR

Existem poucas estratégias de cruzamento. Outras, além da binomial é a estratégia exponencial. Nesta estratégia, o cruzamento é realizado nas variáveis enquanto o número aleatório θ_i for menor ou igual que a probabilidade de cruzamento *CR*. A primeira vez que este número aleatório ultrapassar o valor de *CR*, nenhum cruzamento é executado para o restante das componentes, ou seja:

$$e_i=\begin{cases} d_i, \text{ enquanto } \theta_i \leq CR \\ a_i, \text{ se } \theta_i > CR, \forall\ i=\{i, i+1, \dots, n\} \end{cases} \tag{23}$$

Lin *et al*. (2010), a fim de compreender o papel do cruzamento em ED, fizeram análise teórica e estudo comparativo de cruzamento em ED e projetaram dois novos métodos de cruzamento, chamados cruzamento binomial consecutivo e cruzamento exponencial não consecutivo.

A FIGURA 9 apresenta um exemplo de cruzamento exponencial.

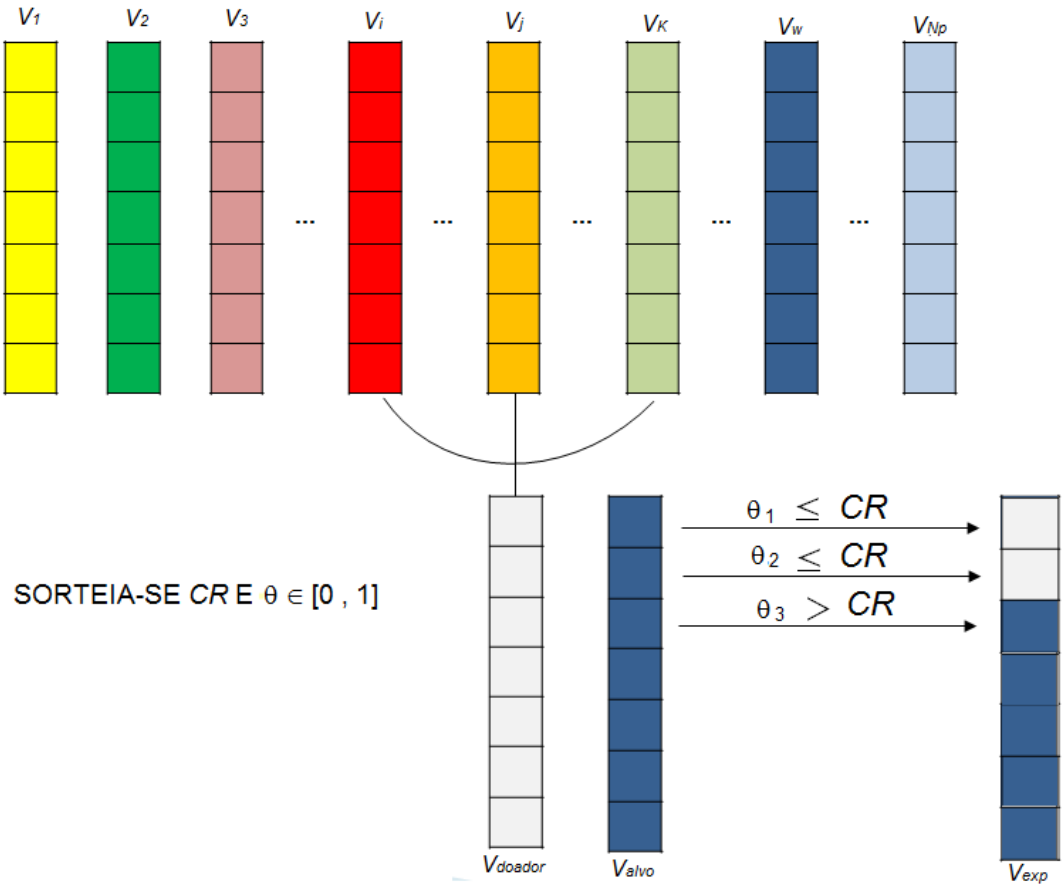

FIGURA 9 – EXEMPLO DO PROCESSO DE CRUZAMENTO EXPONENCIAL
FONTE: AUTOR

2.6.1.5 Seleção

A seleção é feita comparando-se o valor do *fitness* dos V_{exp} e V_{alvo}. Se o valor do *fitness* do vetor V_{exp} for menor (problemas de minimização) que o do V_{alvo}, então o vetor V_{exp} substitui o vetor V_{alvo} na geração seguinte, caso contrário o V_{alvo} permanece na população.

2.6.1.6 Parâmetros

O algoritmo Evolução Diferencial possui três parâmetros: fator de amplificação do vetor diferença (*F*), a taxa de cruzamento *(CR*) e o tamanho da

população (N_p). Uma abordagem amplamente praticada para identificar um bom conjunto de parâmetros para uma classe particular do problema pode ser encontrada nos estudos de Aine *et al.* (2009). O ED original mantém os três parâmetros fixos durante o processo de otimização. No entanto, existe ainda uma falta de conhecimento de como encontrar valores adequados para os parâmetros de controle de ED, para uma determinada função [Liu e Lampinen, 2005]. O fator *F* é um valor real, normalmente no intervalo [0, 2], que controla a amplitude do vetor diferencial e *CR* é um valor real no intervalo [0, 1] que controla a probabilidade de cruzamento. Maiores valores de *F* resultam em maior diversidade na população gerada, e menores valores resultam em convergência, possivelmente para mínimos (máximos) locais.

2.6.1.7 Critério de término

O critério de término de ED pode ser: um número determinado de iterações consecutivas, dentro do qual não há melhoria nas soluções, ou um tempo determinado de CPU (*Central Processing Unit*), ou um número máximo de iterações ($G_{máx}$) ou ainda, quando um número máximo de avaliações de indivíduos (cálculo de *fitness*) é atingido.

2.6.2 Estratégias de Evolução Diferencial

Diferentes estratégias podem ser adotadas em ED dependendo do tipo de problema para o qual o algoritmo é aplicado. As estratégias podem variar de acordo com o vetor a ser perturbado, o número de diferença de vetores considerado para perturbação, as atribuições para os parâmetros e, finalmente, o tipo de cruzamento utilizado [Babu e Jehan, 2003].

A operação de cruzamento pode variar na sua forma de aplicação, podendo ser o binomial ou o exponencial. Na mutação a variação pode ser em relação à escolha de qual indivíduo, por exemplo, qualquer um ou o melhor da geração deve sofrer a modificação para a formação do vetor doador, bem como a quantidade de

indivíduos que são escolhidos para fazer a perturbação. Uma notação adotada para especificar as estratégias pode ser escrita da seguinte forma [Price e Storn, 1997]:

$$\text{ED}/A/B/C \tag{24}$$

Onde:

- *A* especifica o vetor a ser mutado, que pode ser um vetor da população escolhido aleatoriamente, denominado “rand”, ou o vetor de custos mais baixos da população atual, em caso de minimização, chamado de “melhor”;
- *B* indica o número de diferenças ponderadas utilizadas para a perturbação de *A;*
- *C* denota o tipo de cruzamento (exp: exponencial, bin: binomial, etc.).

Em relação aos parâmetros, pesquisas mostram que valores fixos dos parâmetros *F* e *CR* podem facilmente levar à convergência prematura e perda da robustez [Noman e Iba, 2008], [Zamuda *et al.*, 2007]. Para tentar resolver este problema, Zhang, Zhao e Wang (2009) descreveram uma estratégia de ajuste adaptativo, conforme as equações (25) e (26):

$$F = F_0 \; e^{-\beta_1\left(\frac{G}{G_{máx}}\right)} \tag{25}$$

$$CR = CR_0 \, e^{-\beta_2\left(\frac{G}{G_{máx}}\right)} \tag{26}$$

Onde:

- *G* é a geração atual;
- $G_{máx}$ é o número máximo de gerações;
- F_0 e CR_0 são, respectivamente, o valor inicial de fator de escala e a taxa de cruzamento;
- β_1 e β_2 são constantes positivas.

Eiben e Schut (2008) dividem as formas de configuração de parâmetros em dois grandes grupos, conhecidos como:

i. Ajuste de Parâmetros: Nesta forma tenta-se encontrar bons valores de parâmetros antes da execução do algoritmo. O método é ajustado com estes valores, os quais se manterão fixos durante toda a execução;

ii. Controle de Parâmetros: Nesta forma os parâmetros são modificados dinamicamente durante a execução do algoritmo.

Ainda segundo Eiben e Schut (2008), a segunda forma pode ser dividida em três classes:

i. Determinístico: ocorre quando o valor do parâmetro é alterado por alguma regra determinística e, em geral, a heurística é ativada em momentos predeterminados e causa uma mudança também predeterminada sem a utilização de nenhuma informação do processo de busca;

ii. Adaptativo: ocorre quando alguma informação do processo de busca é utilizada para realimentar o mecanismo de variação de parâmetros;

iii. Autoadaptativo: ocorre quando os parâmetros a serem configurados são codificados na representação do indivíduo e passam a sofrer ação dos operadores de variação (mutação e cruzamento).

2.6.3 Evolução Diferencial Discreta

O algoritmo Evolução Diferencial foi proposto inicialmente por Price e Storn (1995) para minimizar funções de espaço contínuo não linear e não diferenciável. O operador aritmético utilizado na operação de mutação em ED é simples, porém a maneira pela qual está definido, torna praticamente impossível a aplicação do ED original a outros espaços [Pampará, *et al.*, 2006]. O operador de mutação do ED original leva à conclusão de que este método só pode ser aplicado no campo dos números reais. Assim, não pode ser usado em problemas de otimização discreta diretamente [Deng *et al.*, 2008; Deng *et al.*, 2009] e se torna impróprio para muitos problemas reais de otimização combinatória que operam no espaço binário [Hota e Pat, 2010].

Em problemas de otimização combinatória, os vetores diferenciais não têm uma correspondência clara com as direções no espaço de busca. Na operação de mutação, a subtração direta de soluções codificadas como vetores de valores inteiros, frequentemente, não geram soluções factíveis e nem representam direções significativas.

Nota-se que cada novo problema de otimização combinatória discreto, resolvido utilizando o algoritmo da Evolução Diferencial Discreta (EDD), gera novas propostas de melhoria para o método EDD. As propostas, em geral, se referem a adaptações nos operadores genéticos, sobretudo sobre o operador de mutação, visando gerar soluções factíveis e maior velocidade de convergência para a solução otimizada.

No algoritmo ED clássico, a principal característica da mutação diferencial é usar informações sobre o que é diferente nos indivíduos da população, de modo a produzir direções e o tamanho do passo da mutação. Entretanto, em problemas de natureza combinatória, a diferença vetorial não tem uma correspondência clara nas direções do espaço de busca. As subtrações de vetores, codificados com números inteiros, não geram soluções viáveis e nem representam direções significativas, uma vez que, os números nas soluções são rótulos arbitrários e não representam nenhuma quantidade numérica. Existem abordagens em que os elementos do vetor diferença são arredondados para o próximo número inteiro válido, de maneira a produzir soluções viáveis [Qian *et al.*, 2009].

Algumas versões discretas de ED, que foram propostas na literatura, são mencionadas a seguir.

Os autores do algoritmo ED, Price e Storn (2005), visando à adaptação do mesmo para o domínio discreto, propuseram uma abordagem para o operador de mutação diferencial, que se utiliza de matriz de permutação. Uma matriz de permutação P é a matriz que mapeia a permutação de um dado vetor em outro. Se V_j e V_k são dois vetores escolhidos de forma aleatória da população, a matriz de permutação que mapeia V_j em V_k satisfaz a relação:

$$V_j = P\ V_k \tag{27}$$

A matriz de permutação é dada pela permutação das linhas da correspondente matriz Identidade, realizando assim a permutação necessária dos

elementos de V_k. Pode-se dizer, portanto, que a matriz de permutação "leva" V_k em V_j. No contexto da evolução diferencial, essa matriz é considerada a "diferença" obtida de duas soluções candidatas. A equação análoga à equação da mutação diferencial (Eq. 11) é, então, escrita como:

$$V_{doador} = P_\eta \, V_i \tag{28}$$

onde V_i é o vetor da população, escolhido aleatoriamente, a sofrer uma perturbação e P_η é a matriz de permutação modificada pelo parâmetro escalar η, significando aqui, uma probabilidade de se utilizar uma parcela da permutação representada pela matriz de permutação original *P*. Portanto, o método se utiliza de uma matriz de permutação que aplica algumas permutações ao vetor base V_i , aleatoriamente selecionadas do conjunto de permutações de *P*. Esta abordagem é tão somente aplicável aos problemas combinatórios baseados em permutações.

Prado *et al.* (2010) também propuseram uma abordagem meta-heurística para a Evolução Diferencial para a otimização discreta, definindo a diferença entre duas soluções candidatas como uma lista de movimentos no espaço de busca, e assim preservando o mecanismo de busca em domínios discretos. Os movimentos da lista são aplicados a outras soluções candidatas da população, como é requerido pelo operador de mutação diferencial. O método foi aplicado em problemas do caixeiro viajante e no problema das *N*-Rainhas. Tasgetiren, Suganthan e Pan (2010) também apresentaram uma adaptação do algoritmo EDD para o problema do caixeiro viajante.

Krauser *et al.* (2013a) compararam os seguintes métodos de computação evolucionária: Algoritmos Genéticos, uma versão binária da Evolução Diferencial (Krauser e Lopes 2012), o Algoritmo Colônia de Abelhas Artificiais e o Algoritmo do Morcego, que bordaram Problemas Binários de Múltiplas Mochilas, onde concluíram que a versão binária do Algoritmo Evolução Diferencial foi o melhor método entre os analisados.

Krauser (2014) propôs uma modelagem matemática para um problema real de rede de distribuição de derivados de petróleo. Programação Linear Inteira Mista e as meta-heurísticas Evolução Diferencial Binário (Krauser e Lopes 2012) e Discretizada (Krauser *et al.* 2013b), que se utiliza de uma função sigmóide no

processo de discretização, são utilizados como métodos de resolução do modelo matemático proposto.

Outras aplicações do EDD no domínio discreto podem ser citadas: *scheduling flowshop* [Onwubolu e Davendra, 2006; Pan, Tasgetiren e Liang, 2008; Pan, Wang e Qian, 2009; Qian *et al*., 2009; Wang, *et al*., 2010]; otimização de projetos de engenharia com variáveis mistas [Liao, 2010].

3. METODOLOGIA

O algoritmo ED, na sua forma clássica, não abrange problemas de natureza discreta. As adaptações do algoritmo de ED apresentadas na literatura para otimização combinatória, basicamente não abordam problemas de localização de instalações.

Sendo assim, o trabalho apresenta um novo operador de mutação para o algoritmo ED, para que o mesmo possa ser aplicado em problemas de otimização combinatória, mais especificamente em problemas de *P*-Medianas e de Máxima Cobertura.

3.1 ALGORITMO HÍBRIDO PROPOSTO

O algoritmo ED adaptado e combinado com o algoritmo Busca Tabu, que é utilizado para podar o espaço de busca contribuindo para uma convergência mais rápida e precisa, formam o algoritmo híbrido proposto denominado EDBT. O algoritmo Busca Tabu atua diretamente no novo operador de mutação proposto para o algoritmo Evolução Diferencial.

No algoritmo, o valor do *fitness* corresponde, para os problemas de *P*-Medianas e Máxima Cobertura, aos respectivos valores das funções objetivos determinadas nas Equações (1) e (6).

O EDBT aborda os problemas de localização *P*-Medianas e Máxima Cobertura. Pode-se classificar, de acordo com Dubke (2006), os problemas de localização de instalações, abordados neste trabalho, como modelos planos, pois se utiliza das coordenadas cartesianas (X,Y) dos vértices, e discretos, pois as instalações devem estar localizadas nos vértices do problema em estudo (ver seção 2.1.1). Trata-se ainda a localização como um problema não capacitado ou com capacidade ilimitada.

O algoritmo EDBT é construído a partir do algoritmo ED. Na operação de mutação do algoritmo ED se propôs uma adaptação, que se utiliza da componente Lista Tabu, do algoritmo BT, formando assim o modelo de hibridização, desenvolvido

no EDBT. Os algoritmos ED e BT são executados simultaneamente e atuam como equipes trocando informações entre si, havendo uma combinação integrada, ou seja, os algoritmos envolvidos possuem uma técnica, como um componente subordinado, encaixada dentro da outra. De acordo com Yen *et al*. (1998), seria classificado na categoria: Algoritmo com operadores adicionais (ver seção 2.2.4).

Ressalta-se ainda que o algoritmo EDBT gera na operação de mutação vetores factíveis, ou seja, vetores que pertencem ao espaço de busca, não havendo necessidade de se utilizar artifícios de correção.

A seguir, no QUADRO 2 , apresentam-se as variáveis e as componentes definidas no algoritmo proposto.

QUADRO 2- DESCRIÇÃO DAS VARIÁVEIS E COMPONENTES DO EDBT

Variáveis	**Descrição**
$G_{máx}$	Número máximo de iterações do algoritmo.
N_p	Número de vetores da população $G = \{V_1, V_2, ..., V_{Np}\}$.
N_{med}	Número de medianas a ser determinada.
N_v	Número total de vértices do problema.
(X_{ij}, Y_{ij})	Coordenada cartesiana da *i*-ésima mediana do *j*-ésimo vetor da população $G = \{V_1, V_2, ..., V_{Np}\}$.
λ e CR	Valores reais compreendidos entre 0 e 1.
F	Valor real compreendido entre 0 e 1,5.
V_{melhor}	Vetor que apresenta melhor *fitness* da iteração corrente.
δ	Número inteiro positivo que indica a quantidade de medianas do V_{melhor} que irão sofrer a mutação.
V_{alvo}	Vetor alvo pertencente à população $G = \{V_1, V_2, ..., V_{Np}\}$.
V_d	Vetor doador gerado na operação de mutação do algoritmo.
L	Valor inteiro que representa o limite máximo de vezes que um vértice participará dos movimentos de mutação antes de entrar na *Lista_Tabu*.

Continua

QUADRO 2- DESCRIÇÃO DAS VARIÁVEIS E COMPONENTES DO EDBT

Conclusão

TB(i)	Valor inteiro que representa o número de vezes que o vértice *i* participa dos movimentos de mutação.
Lista_Tabu	Lista que armazena os últimos *K* vértices que foram designados para participar da operação de mutação mais de *TB* vezes.
K	Representa a dimensão da *Lista_Tabu*, ou seja, a quantidade de iterações, do movimento na *Lista_Tabu.*

Os passos do algoritmo EDBT, estão descritos a seguir:

P1: Atribuir aleatoriamente valores para os parâmetros $G_{máx}$, N_p, δ, *CR*, *L* e *K*, dentro dos intervalos de definição destes;

P2: Gerar aleatoriamente uma população inicial contendo N_p vetores factíveis.

P3: Avaliar cada vetor da população inicial de acordo com a função objetivo, obtendo-se assim o valor *fitness*, correspondente ao problema abordado.

P4: Identificar na população V_{melhor} ;

P5: Escolher de forma ordenada o V_{alvo}, ou seja, na *i*-ésima operação de mutação da iteração o V_{alvo} escolhido é o *i*-ésimo vetor da população.

P6: A seguir sorteiam-se:

- dois vetores V_i e V_j da população, de modo que V_i, V_j, V_{alvo} e V_{melhor} sejam distintos entre si;
- uma componente do vetor V_i e uma do vetor V_j cujas coordenadas cartesianas são respectivamente (X_{ni}, Y_{ni}) e (X_{mj}, Y_{mj});
- δ medianas do V_{melhor}, que participarão da operação de mutação, formando um subconjunto *A* não vazio;
- um valor real para *F* compreendido entre 0 e 1,5;
- um valor real para λ compreendido entre 0 e 1.

P7: Obter as coordenadas cartesianas (X ,Y) pertencentes a um ponto P do espaço contínuo da seguinte forma: $\in$

$$X = F \cdot ((1 - \lambda) \cdot X_{ni} + \lambda \cdot X_{mj}) \quad (29)$$

$$Y = F \cdot ((1 - \lambda) \cdot Y_{ni} + \lambda \cdot Y_{mj}) \quad (30)$$

P8: Determinar o vértice P_0 do problema, mais próximo do ponto P tal que:

- $P_0 \notin V_{melhor}$;

- $P_0 \notin$ *Lista_Tabu*;

P9: Substituir a mediana a_i, selecionada do subconjunto A, pelo vértice P_0.

P10: Atualizar *Lista_Tabu*:

- $TB(i) = TB(i) + 1$ $\forall$ vértice $i \in$ *Lista_Tabu*.

- se $TB(i) > |K|$, então o vértice i sai da *Lista_Tabu* podendo novamente participar da operação de mutação.

- caso o número de vezes que P_0 tenha sido escolhido, nas iterações anteriores, para participar da mutação seja maior que L então o vértice P_0 é incluído na *Lista_Tabu.*

P11: Obter o vetor doador (V_{doador}) após as δ medianas, pertencentes ao subconjunto A, sofrerem a mutação.

P12: Após a operação de mutação realizar o cruzamento binomial, descrito anteriormente (ver Seção 3), entre o V_{doador} e o V_{alvo}, com uma probabilidade de cruzamento CR definido no Passo 1, resultando no vetor experimental (V_{exp}).

P13: Aplicar a seleção gulosa, ou seja, a seleção é feita comparando o valor do *fitness* dos vetores V_{exp} e V_{alvo}. Se o valor do *fitness* do V_{exp} for menor (se o problema for de minimização) que o do V_{alvo}, então o V_{exp} substitui o V_{alvo} na população seguinte.

P14: Se todos os vetores da população foram designados como V_{alvo}, então uma iteração é concluída, ir para o Passo 15, caso contrário ir para o Passo 3.

P15: Se critério de parada não for satisfeito: voltar para P3 e $G_{máx} = G_{máx} + 1$, caso contrário: **FIM**.

A FIGURA 10 mostra um exemplo bidimensional ilustrando os vetores que participam da geração de uma componente do vetor doador (V_{doador}).

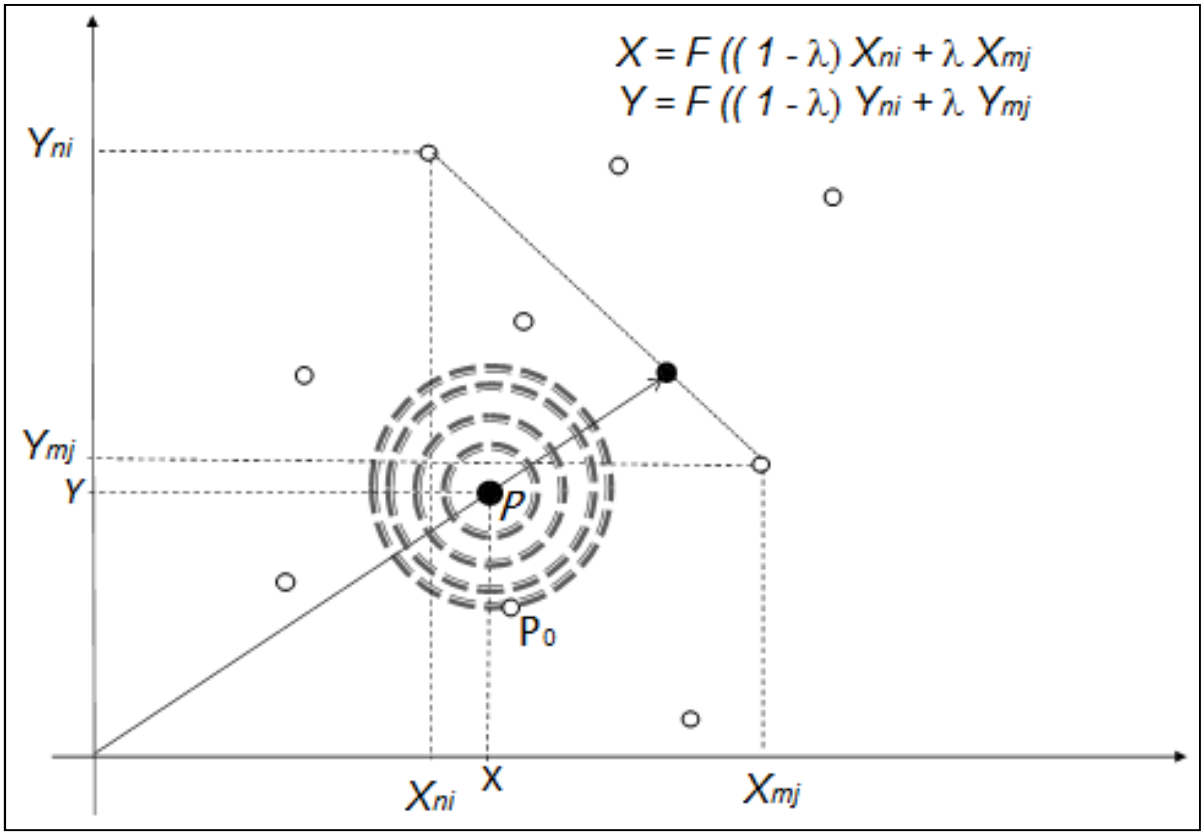

FIGURA 10- PROCESSO DE GERAÇÃO DE UMA COMPONENTE DO VETOR DOADOR (V_{doador}).
FONTE: AUTORA

No operador de mutação, as operações entre vetores, codificados com números inteiros, geralmente não geram soluções viáveis ao se multiplicar, por exemplo, um vetor por um escalar F ou λ compreendidos em um intervalo contínuo. A proposta de se utilizar as coordenadas cartesianas dos vértices neste operador tem o propósito de gerar novos vértices, no espaço contínuo, que estejam localizados no espaço de busca. O passo P6, do algoritmo proposto, retorna ao

espaço discreto ao identificar o vértice do problema mais próximo do gerado no espaço contínuo, apresentando assim soluções viáveis.

A configuração de parâmetros no algoritmo é uma etapa importante para se obter um desempenho adequado, além disso, a configuração ideal destes parâmetros depende da função que se quer otimizar.

O algoritmo Evolução Diferencial clássico é um algoritmo evolutivo simples que opera com poucos parâmetros. Com o propósito em se manter esta característica do ED, procuraram-se, neste trabalho, regras determinísticas para os parâmetros *L* e *K* de maneira que as mesmas permitam que o algoritmo proposto apresente bom desempenho no que se refere à qualidade de soluções e se adapte de acordo com o problema que se está tratando, sem interferência do usuário.

Após a realização de diversos testes computacionais aplicados nas instâncias utilizadas neste trabalho, descritas no próximo capítulo, estabeleceu-se a seguinte regra para os parâmetros *L* e *K*:

$$L = \left\lfloor \frac{G_{\text{máx}}}{N_p} \right\rfloor \tag{31}$$

$$K = \left\lfloor \frac{G_{\text{máx}} - \text{Iteração}}{2} \right\rfloor \tag{32}$$

Ressalta-se ainda, que os parâmetros *F* e λ são parâmetros que o algoritmo sorteia aleatoriamente dentro de um intervalo real estabelecido e desta forma, também são livres da ação do usuário.

3.4.1 Fluxograma do algoritmo EDBT

A FIGURA 11, a seguir, apresenta o fluxograma do algoritmo EDBT proposto neste trabalho. O operador mutação foi detalhado pois é o operador que recebeu a proposta de adaptação do algoritmo Evolução Diferencial para atuar em problemas discretos.

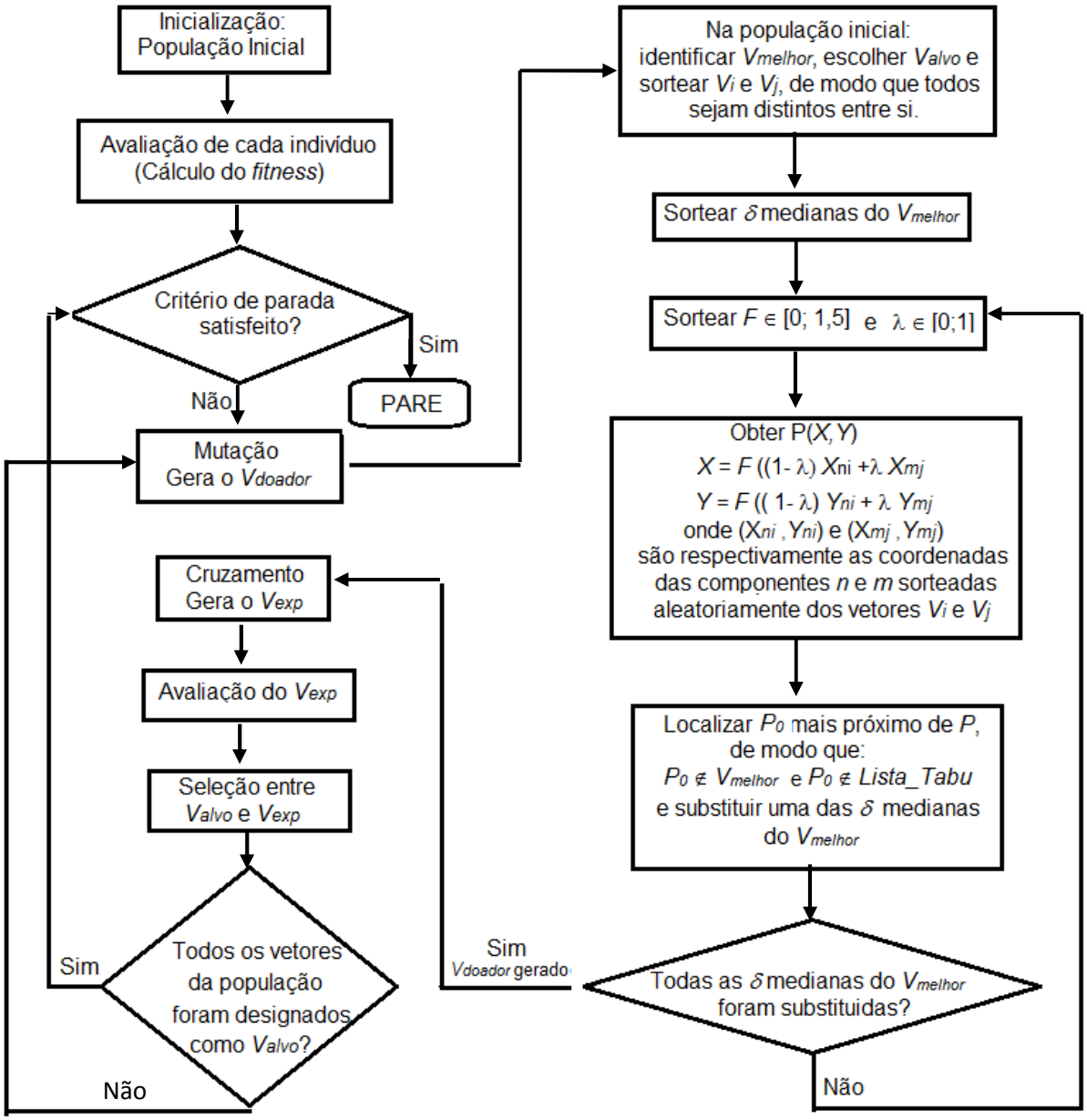

FIGURA 11- FLUXOGRAMA EDBT
FONTE: AUTORA

A seguir, apresenta-se um exemplo numérico para melhor entendimento do algoritmo proposto.

3.4.2 Exemplo Ilustrativo para um PPM

Segue um exemplo ilustrativo de uma iteração do algoritmo EDBT para um PPM.

Seja um problema com 10 vértices onde se quer localizar 3 medianas, de modo que a soma total das distâncias dos outros vértices até a mediana mais próxima seja mínima. Considera-se um problema sem restrição de capacidade e com distância euclidiana entre os vértices.

Apresentam-se a seguir as Tabelas 1 e 2 com as coordenadas cartesianas dos 10 vértices e a matriz de distância euclidiana (MD) entre os mesmos.

TABELA 1 – COORDENADAS CARTESIANAS

PONTO	X	Y	PONTO	X	Y
1	20	38	6	47	47
2	43	35	7	17	19
3	25	9	8	38	25
4	35	39	9	12	43
5	47	30	10	12	6

TABELA 2 – MATRIZ DE DISTÂNCIAS

$$MD=\begin{bmatrix} 0,00 & 23,19 & 29,43 & 15,03 & 28,16 & 28,46 & 19,24 & 22,20 & 9,43 & 32,98 \\ 23,19 & 0,00 & 31,62 & 8,94 & 6,40 & 12,65 & 30,53 & 11,18 & 32,02 & 42,45 \\ 29,43 & 31,62 & 0,00 & 31,62 & 30,41 & 43,91 & 12,81 & 20,62 & 36,40 & 13,34 \\ 15,03 & 8,94 & 31,62 & 0,00 & 15,00 & 14,42 & 26,91 & 14,32 & 23,35 & 40,22 \\ 28,16 & 6,40 & 30,41 & 15,00 & 0,00 & 17,00 & 31,95 & 10,30 & 37,34 & 42,44 \\ 28,46 & 12,65 & 43,91 & 14,42 & 17,00 & 0,00 & 41,04 & 23,77 & 35,23 & 53,91 \\ 19,24 & 30,53 & 12,81 & 26,91 & 31,95 & 41,04 & 0,00 & 21,84 & 24,52 & 13,93 \\ 22,20 & 11,18 & 20,62 & 14,32 & 10,30 & 23,77 & 21,84 & 0,00 & 31,62 & 32,20 \\ 9,43 & 32,02 & 36,40 & 23,35 & 37,34 & 35,23 & 24,52 & 31,62 & 0,00 & 37,00 \\ 32,98 & 42,45 & 13,34 & 40,22 & 42,44 & 53,91 & 13,93 & 32,20 & 37,00 & 0,00 \end{bmatrix}$$

1ª Iteração:

P1: Define-se os parâmetros: $G_{máx} = 20$; $N_p = 5$; $\delta = 1$; $CR = 0,75$; $L = G_{máx}/N_p = 4$ e $K = \lfloor (G_{máx} - \text{Iteração})/2 \rfloor = \lfloor (20 - 1)/2 \rfloor = 9$.

P2: Gerar a população inicial:

9	5	2	V_1
1	2	3	V_2
4	3	6	V_3
8	7	5	V_4
6	9	1	V_5

P3: Calcular o *fitness* de cada vetor :

fitness (V_1) = 133,25

fitness (V_2) = 74,76

fitness (V_3) = 102,79

fitness (V_4) = 108,21

fitness (V_5) = 102,98

Obs: O *fitness* do vetor é o valor da soma das distâncias entre os vértices e as medianas mais próximas definidas no vetor. Por exemplo: as medianas do V_2 são 1, 2 e 3. Assim temos que:
fitness (V_2) = min{d(4,1), d(4,2), d(4,3)} + min{d(5,1), d(5,2), d(5,3)} + min{d(6,1), d(6,2), d(6,3)} + min{d(7,1), d(7,2), d(7,3)} + min{d(8,1), d(8,2), d(8,3)} + min{d(9,1), d(9,2), d(9,3)} + min{d(10,1), d(10,2), d(10,3)} = 8,94 + 6,40 + 12,65 + 12,81 + 11,18 + 9,43 + 13,34 = 74,76

P4: O $V_{alvo} = V_1$ cujo *fitness* é igual a 133,25 (escolha ordenada da população).

P5: O $V_{melhor} = V_2$ cujo *fitness* é igual a 74,76.

P6: Sorteiam-se:

- dois vetores da população: V_4 e V_5;

- uma componente de cada vetor sorteado: 5 do V_4 e 9 do V_5 ;
- uma componente, pois $\delta = 1$, a_i do V_{melhor}. Logo o conjunto A formado pelas componentes do V_{melhor} que serão substituídas na operação de mutação. Pode-se supor, após sorteio, que A={3}, ou seja a componente sorteada é igual a 3.
- $F = 0,67$
- $\lambda = 0,23$

P7: Calcular P(X,Y).

$$X = F \cdot ((1 - \lambda) \cdot X_{n1} + \lambda \cdot X_{m2}) = X = 0,67 \cdot ((1 - 0,23) \cdot 47 + 0,23 \cdot 12) = 5,44$$
$$Y = F \cdot ((1 - \lambda) \cdot Y_{n1} + \lambda \cdot Y_{m2}) = Y = 0,67 \cdot ((1 - 0,23) \cdot 30 + 0,23 \cdot 43) = 6,60$$
$$P = (5,44; 6,60)$$

P8: O vértice P_0 mais próximo de P é o 10. Como o número de vezes que o vértice 10 foi escolhido (que no momento é igual a 1) é menor ou igual a $L = 4$ e $10 \notin V_{melhor}$, então o vértice 10 poderá participar da operação de mutação.

P9: Substituir a mediana 3 do V_{melhor}, sorteada no Passo 3, pelo vértice $P_0 = 10$, determinado no Passo 6: (1, 10, 2).

P10: A $Lista\_Tabu = \varnothing$, pois nesta primeira iteração ainda não há vértices que participaram da operação de mutação um número maior que 4 (pois $L = 4$).

P11: Como $\delta = 1$ não há mais componentes do V_{melhor} a serem mutadas e desta forma $V_{doador} = (1, 10, 2)$.

P12: Realizar cruzamento binomial, sorteando os valores para $\theta 1$, $\theta 2$ e $\theta 3$, resultando o V_{exp}.

V_{alvo}	V_{doador}			V_{exp}
9	1	$\theta_1 = 0,52$	$\leq CR = 0,75 \rightarrow$	9
5	2	$\theta_2 = 0,88$	$> CR = 0,75 \rightarrow$	2
2	10	$\theta_3 = 0,76$	$> CR = 0,75 \rightarrow$	10

P13: Realizar a seleção gulosa: Temos que *fitness* (V_{exp}) = 75,88 e *fitness* (V_{alvo}) = 133,25, logo *fitness* (V_{exp}) < *fitness* (V_{alvo}) e desta forma o $V_{alvo} = V_1$ é substituído pelo V_{exp} na população.

A nova população será:

9	2	10	V_1
1	2	3	V_2
4	3	6	V_3
8	7	5	V_4
6	9	1	V_5

P14: Existem vetores da população que não foram designados como V_{alvo}, então ir para o Passo 3.

P3: Calcular o *fitness* de cada vetor :

fitness (V_1) = 75,88

fitness (V_2) = 74,76

fitness (V_3) = 102,79

fitness (V_4) = 108,21

fitness (V_5) = 102,98

P4: O $V_{melhor} = V_2$ (apresenta o melhor valor do *fitness*)

P5: $V_{alvo} = V_2$ cujo *fitness* é igual a 74,76 (escolha ordenada da população).

Os Passos seguintes devem ser seguidos até que se tenha no Passo 14 a condição de que todos os vetores da população tenham sido designados como V_{alvo}, seguindo então para o Passo 15.

Passo 15: Critério de parada não foi satisfeito, pois $G_{máx} = 1 < 20$. Então atualizar $G_{máx} = G_{máx} + 1 = 1 + 1=2$ e iniciar a Segunda Iteração.

O algoritmo EDBT segue os mesmos passos ao ser aplicado a Problemas de Máxima Cobertura, tendo como diferencial a função objetivo para se determinar o *fitness* dos vetores.

No capítulo seguinte, apresentam-se os resultados obtidos utilizando o algoritmo EDBT, com as adaptações apresentadas, comparadas com algumas instâncias propostas na literatura e outras geradas aleatoriamente com distribuição uniforme.

4 TESTES COMPUTACIONAIS E ANÁLISE DOS RESULTADOS

Os algoritmos abordados foram programados em *Visual Basic*, versão 2010, e para o desenvolvimento computacional foi utilizado um computador Sony com processador Intel Core i5 com 640 GB de HD, 6 GB de RAM, 2.0 GHz e sistema operacional Windows 7.

Com o propósito de avaliar o desempenho do algoritmo proposto neste trabalho, para os problemas em relação à qualidade das soluções obtidas e o tempo computacional necessário à obtenção da melhor solução, efetuaram-se 10 (dez) simulações para cada problema teste, adotando-se como critério de parada o número máximo de iterações.

As TABELAS 3, 4, 5, 6 e 7 apresentam os resultados computacionais das instâncias descritas nas próximas subseções. Os parâmetros, definidos de forma empírica, para cada instância testada encontram-se no ANEXO 1 e as principais legendas dos símbolos utilizados nestas tabelas estão descritas a seguir:

- *n*: número de vértices do problema;
- *p*: número de medianas (facilidades);
- *S*: raio de cobertura;
- S_ÓTIMA: solução ótima do modelo matemático, descrito por Christofides (1975) para PPM e Church e ReVelle (1974) para PMC;
- C (%): porcentagem da demanda coberta pelas *p* facilidades;
- S_LORENA: melhor solução apresentada por Lorena *et. al* (2001) entre as 100 simulações efetuadas para cada instância;
- S_EDBT: melhor solução, encontrada pelo algoritmo aqui proposto, entre as 10 simulações efetuadas para cada instância;
- D_p: desvios percentuais apresentados pelos métodos abordados que foram calculados da seguinte forma:

$$D_p = 100.(\text{ S\_EDBT} - \text{S\_REFERÊNCIA})/\text{ S\_REFERÊNCIA}. \qquad (33)$$

Onde S_REFERÊNCIA equivale a melhor solução do método utilizado para a comparação dos resultados obtidos pelo método aqui proposto.

- TEMPO: tempo médio computacional dos métodos aplicados, entre as simulações efetuadas, em segundos.

4.1 TESTES E RESULTADOS PARA OS PPM

Para os testes realizados em PPM, utilizaram-se dois grupos de instâncias:

- Primeiro grupo de testes para PPM

O primeiro grupo, denominado aqui por instâncias Lorena_*P*-Medianas e disponíveis em: <http://www.lac.inpe.br/~lorena/ArsigIndex.html>, apresenta 20 (vinte) instâncias, contendo 324, 818 e 3282 vértices, cujos resultados foram comparados com os obtidos por:

— Lorena *et al.* (2001), onde o autor integra a alguns SIGs (Sistemas de Informações Geográficas) a implementação de uma abordagem recente da heurística Lagrangeana/*Surrogate* que tem se mostrado eficiente em diversas classes de problemas de Otimização Combinatória.

— Vasconcelos *et al.* (2010), que realizou um estudo comparativo entre aplicações das meta-heurísticas Algoritmo Genético com Busca Local Adaptativo (AG) baseado no método GRASP (*Greedy Randomized Adaptive Search Procedure*), o Jumping Frog Optimization (JFO) ou Algoritmo de Otimização por Saltos de Rãs e o Algoritmo Genético descrito em Bezerra (2008) para a solução do Problema das *P*-Medianas.

- Segundo grupo de testes para PPM

Diante da dificuldade de se encontrar instâncias na literatura que forneçam as coordenadas cartesianas dos vértices, o segundo grupo, contendo 30 (trinta) instâncias, foi gerado aleatoriamente, com distribuição uniforme, com no mínimo 100 e máximo 600 vértices. Para as instâncias em questão, os resultados obtidos com a metodologia proposta foram comparados com as soluções obtidas com o algoritmo

Teitz & Bart. As instâncias geradas estão disponíveis em: http://paginapessoal.utfpr.edu.br/durski.

A TABELA 3 apresenta os resultados obtidos para as instâncias, do primeiro grupo (Lorena_*P*-Medianas), testadas para o problema das *P*-Medianas. Neste grupo de testes o D_p é calculado tomando-se S_REFERÊNCIA equivalente a S_LORENA.

TABELA 3 – RESULTADOS DOS TESTES COMPUTACIONAIS EM RELAÇÃO A LORENA *ET AL*. (2001) PARA PROBLEMAS DE *P*-MEDIANAS

n	*p*	LORENA *et al*. (2001)		EDBT		
		S_LORENA	TEMPO (s)	S_ EDBT	D_p (%)	TEMPO (s)
324	5	122518,00	4,72	122518,00	0	0,80
	10	79256,30	7,30	79256,30	0	1,50
	20	54533,11	7,33	54505,34	-0,05	5,41
	50	32101,50	7,65	32286,29	0,58	23,09
	108	19683,61	7,84	18892,30	-4,02	40,00
818	5	605855,81	102,66	605855,81	0	9,22
	10	385371,44	97,48	384341,82	-0,27	12,67
	20	251717,77	60,39	251713,09	-2E-03	29,39
	50	149251,13	43,73	147204,81	-1,37	69,30
	100	98992,31	57,93	99050,20	0,06	165,00
	150	77440,57	66,19	76918,90	-0,68	258,06
	272	50086,61	85,58	49120,03	-1,93	433,62
3282	5	6381119,00	1699,88	6381119,00	0	168,99
	10	3914249,75	1548,43	3914166,45	-2E-03	179,30
	20	2350502,50	1520,00	2350495,07	-3E-04	357,36
	50	1308957,25	1106,45	1294934,00	-1,07	663,08
	100	841380,81	954,24	824553,19	-2	715,68
	500	332954,84	1530,44	326295,74	-2	1408,00
	1000	194813,50	1606,07	188969,10	-3	2023,65
	1141	175905,27	1526,76	168869,06	-4	3007,72

Com as informações da TABELA 3, constatou-se que o algoritmo EDBT apresentou em 90% das instâncias, soluções melhores ou iguais às apresentadas por Lorena *et al.* (2001).

Em relação aos tempos computacionais uma comparação exata é difícil, pois os algoritmos foram executados em máquinas diferentes, mas a título de registro, os dados serão apresentados. Lorena *et al*. (2001) utilizaram um microcomputador Pentium MMX 233MHz com 128MB de memória RAM.

O tempo computacional do algoritmo proposto é crescente à medida que se aumentam os dados de entrada. São mostrados a seguir os resultados comparativos entre os métodos de acordo com o número de vértices.

Para as 5 (cinco) instâncias onde n = 324, o algoritmo EDBT obteve 80% de soluções melhores ou iguais as apresentadas por Lorena *et al.* (2001), com exceção do problema onde p = 50. O maior ganho na melhoria da solução, apresentado na instância onde p = 108, foram de 4,02%.

O GRÁFICO 1, a seguir, exibe os vértices que foram designados a participarem da operação mutação do algoritmo EDBT. Observa-se que o EDBT possibilita a participação de grande parte dos vértices do problema para a operação de mutação, vindo a realizar uma busca abrangente no espaço de solução.

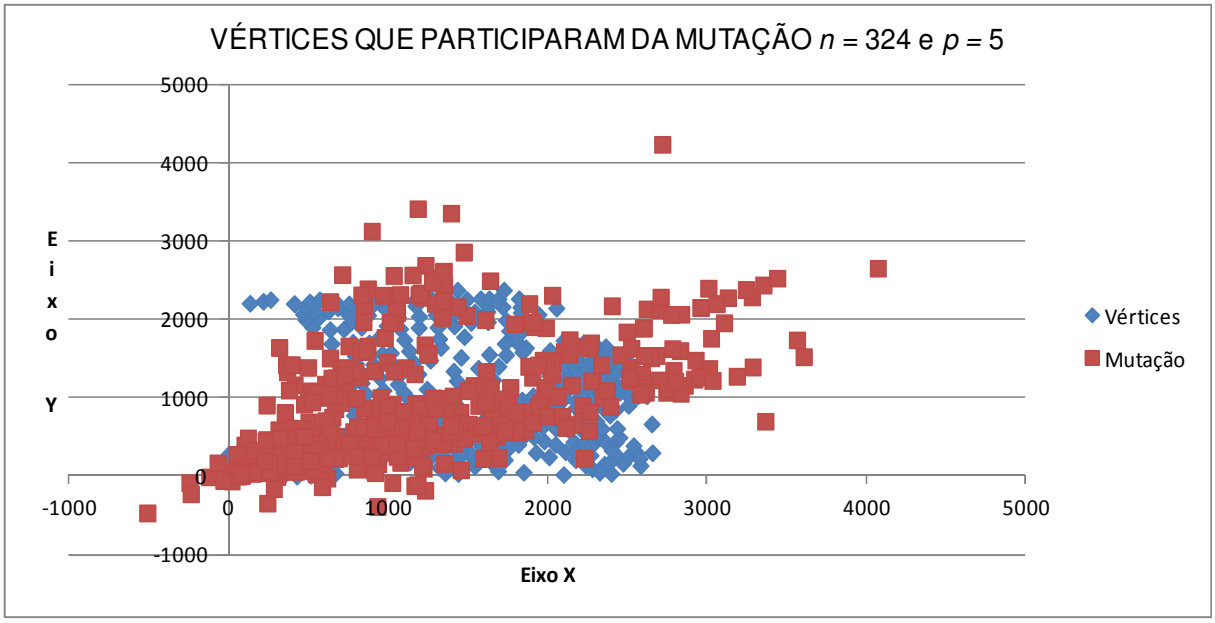

GRÁFICO 1 – ESPAÇO DE BUSCA PARA A OPERAÇÃO DE MUTAÇÃO DO EDBT.

Para as 7 (sete) instâncias onde n = 818, o algoritmo EDBT obteve aproximadamente 85,71% de soluções melhores ou iguais as apresentadas por Lorena *et al.* (2001), com exceção do problema onde p = 100. No problema onde p = 272, o algoritmo EDBT apresentou o maior ganho na melhoria da solução que foi de 2,03%.

Para as 8 (oito) instâncias onde n = 3282, o algoritmo EDBT obteve 100% de soluções melhores ou iguais as apresentadas por Lorena *et al.* (2001).

Os GRÁFICOS 2 e 3 exibem o desempenho da convergência dos testes computacionais efetuados pelo método proposto para as instâncias nas 500 primeiras iterações, onde n = 324, e nas 800 primeiras iterações, onde n = 818, cujos resultados finais estão indicados na (TABELA 3).

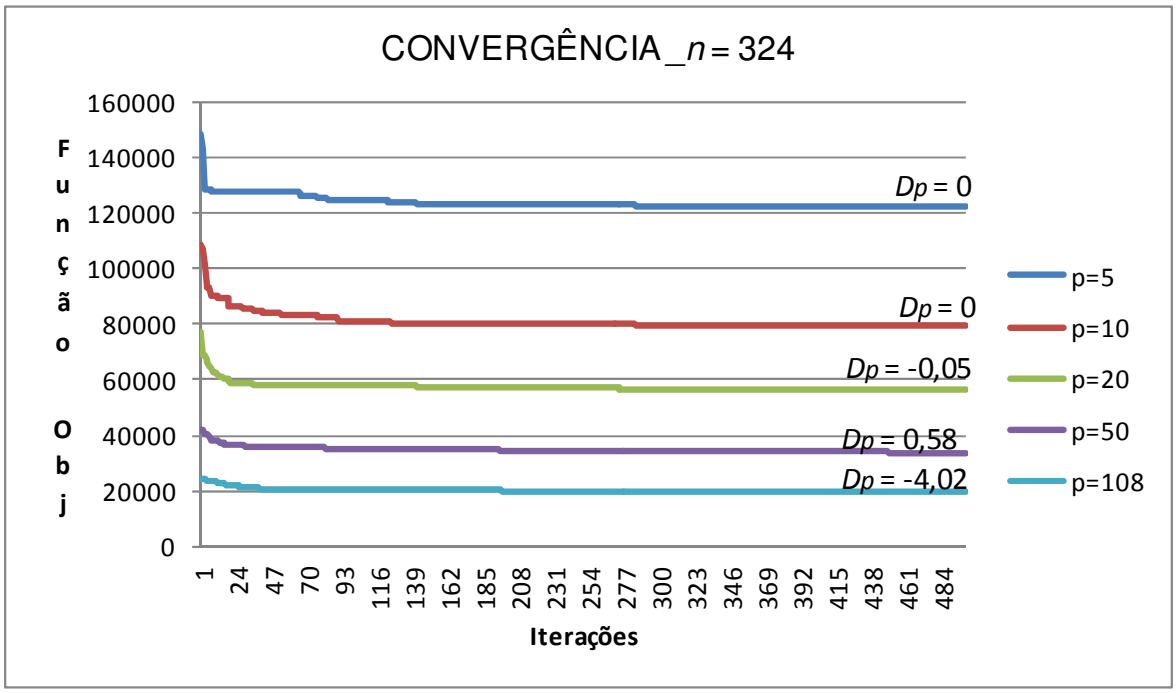

GRÁFICO 2 – DESEMPENHO DE CONVERGÊNCIA DO EDBT (n = 324)

Verifica-se, de acordo com os GRÁFICOS 2 e 3 que o algoritmo proposto apresenta um comportamento homogêneo diante da variação da quantidade de vértices e medianas das instâncias abordadas. As curvas apresentam maiores taxas diferenciais nas iterações iniciais.

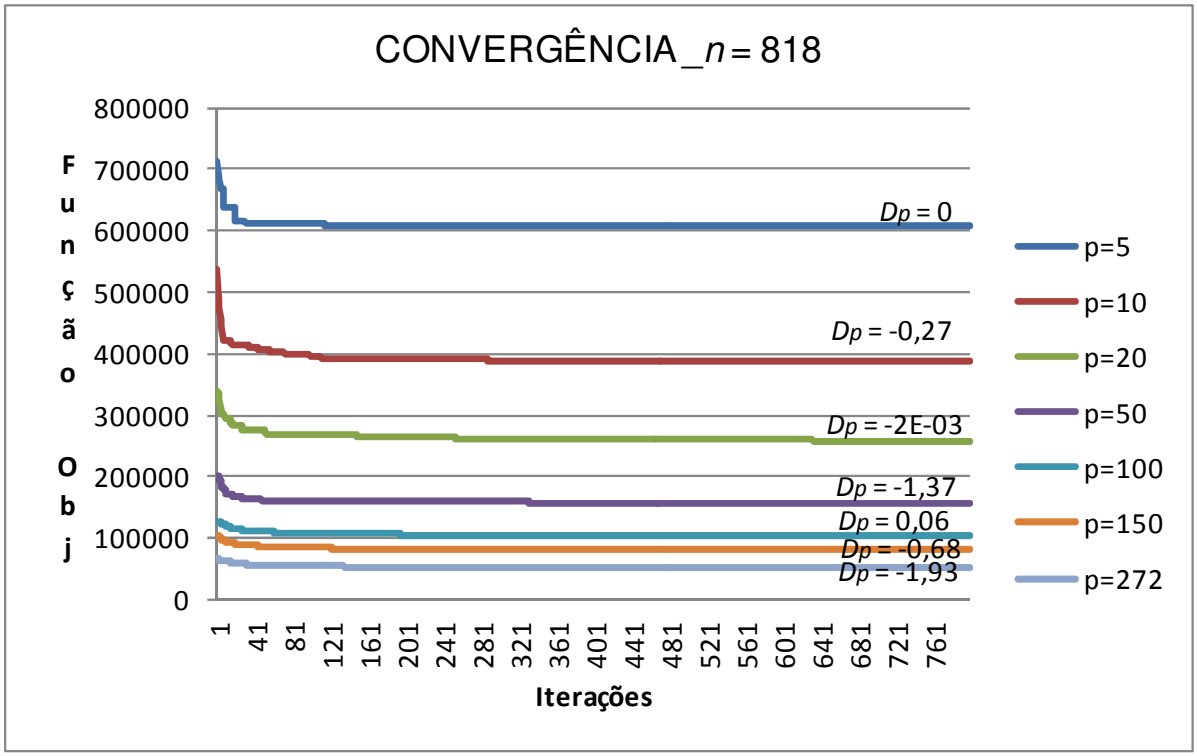

GRÁFICO 3 – DESEMPENHO DE CONVERGÊNCIA DO EDBT (n = 818)

As TABELA 4 e 5, a seguir, apresentam comparações entre os resultados do método EDBT com os resultados obtidos através dos seguintes algoritmos: Genético (AG), Genético com Busca Local (AG_BL) e de Otimização por Salto de Rãs (JFO). Em todos os métodos abordados a média das soluções obtidas, entre as 10 simulações efetuadas, é representado por S_MÉDIA e o D_p é calculado tomando-se S_REFERÊNCIA equivalente a S_ÓTIMA.

Na TABELA 4, os resultados referem-se a instâncias onde n = 324.

Com as informações da TABELA 4, verifica-se que o método EDBT apresentou melhores desvios percentuais, em relação à qualidade das soluções, em 100 % das instâncias em relação aos outros métodos abordados. Ressalta-se ainda que o algoritmo EDBT atingiu a solução ótima para quase todas as variações do valor de p, com exceção da instância onde p = 50.

Vasconcelos *et al.*(2010) implementaram o algoritmo em linguagem de programação C++ Builder-6, e executado 10 vezes para cada mediana em um computador Intel Core 2 Duo com 3 GB de memória RAM e sistema operacional Windows XP. Desta forma, a comparação entre os tempos computacionais dos algoritmos abordados pelos autores e o EDBT não será apresentada, pois os testes foram realizados em diferentes máquinas.

TABELA 4 - RESULTADOS DOS TESTES COMPUTACIONAIS EM RELAÇÃO A VASCONCELOS *ET AL*. (2010) PARA PROBLEMAS DE *P*-MEDIANAS ONDE *n* = 324

p	S_ÓTIMA	ALG	S_MELHOR	S_MÉDIA	TEMPO (s)	D_p (%)
5	122518	AG	127920,00	128554,00	16,22	4,41
		AG_BL	127078,00	129894,00	371,34	3,72
		JFO	126721,00	128556,00	41,44	3,43
		EDBT	122518,00	122518,00	0,80	0
10	79256,36	AG	82855,30	85610,00	14,50	4,54
		AG_BL	82447,00	800976,00	681,00	4,03
		JFO	80463,00	80795,00	26,27	1,52
		EDBT	79256,36	80114,82	1,50	0
20	54533,11	AG	60109,00	61954,00	15,68	10,22
		AG_BL	55573,00	54876,00	3104,4	1,91
		JFO	55428,00	57967,00	581,00	1,64
		EDBT	54533,11	54569,52	5,43	0
50	32101,50	AG	38657,00	39793,00	19,60	20,42
		AG_BL	33496,00	32764,00	7125,00	4,34
		JFO	33396,00	33549,00	1756,00	4,03
		EDBT	32399,29	32511,92	19,22	0,93

Em relação aos resultados de S_MÉDIA, o algoritmo EDBT apresentou melhor média entre as simulações em 100% das instâncias.

Na TABELA 5, os resultados referem-se a instâncias onde *n* = 818.

TABELA 5 - RESULTADOS DOS TESTES COMPUTACIONAIS EM RELAÇÃO A VASCONCELOS *ET AL*. (2010) PARA PROBLEMAS DE *P*-MEDIANAS ONDE *n* = 818

p	S_ÓTIMA	ALG	S_MELHOR	S_MÉDIA	TEMPO (s)	*Dp* (%)
5	605856,1	AG	637874,00	647937,00	18,30	5,28
		AG_BL	628739,00	635784,00	15353,00	3,78
		JFO	619745,00	627694,00	8294,00	2,29
		EDBT	605856,1	607474,01	9,22	0
20	251717,8	AG	310971,00	329524,00	23,66	23,54
		AG_BL	306954,00	326465,00	14941,00	21,94
		JFO	288167,00	637594,00	212,55	14,48
		EDBT	251717,8	252521,60	29,39	0

Com as informações da TABELA 5, verifica-se que o método EDBT apresentou melhores desvios percentuais, em relação à qualidade das soluções, nas 2 (duas) instâncias em relação aos outros métodos abordados. Ressalta-se

ainda que o algoritmo EDBT atingiu a solução ótima para todas as variações do valor de *p*. Em relação aos resultados de S_MÉDIA, o algoritmo EDBT apresentou melhor média nas 2 (duas) instâncias.

A TABELA 6, a seguir, apresenta os resultados obtidos para as instâncias do segundo grupo, geradas aleatoriamente com uma distribuição uniforme, testadas para o PPM. Para este grupo de instâncias o D_p é calculado tomando-se S_REFERÊNCIA equivalente a S_ÓTIMA, obtida com o modelo matemático apresentado por Christofides (1975) que foi resolvido pelo *software* LINGO[1] 13.0, cujos tempos computacionais encontram-se no ANEXO 2 deste trabalho.

TABELA 6 - RESULTADOS DOS TESTES COMPUTACIONAIS COMPARADOS COM TEITZ & BART PARA INSTÂNCIAS GERADAS PARA PROBLEMAS DE *P*-MEDIANAS

n	*p*	LINGO	TEITZ & BART			EDBT		
		S_ÓTIMA	S_MELHOR	TEMPO (s)	*Dp* (%)	S_MELHOR	TEMPO (s)	*Dp* (%)
100	5	351,46	351,46	0,05	0	351,46	0,43	0
	10	219,23	219,23	0,31	0	219,23	0,65	0
	20	130,97	130,97	0,68	0	130,97	0,72	0
	33	86,88	87,29	2,28	0,47	86,88	2,32	0
200	5	3315,98	3315,98	0,18	0	3315,98	0,62	0
	10	2173,28	2173,28	1,06	0	2173,28	2,42	0
	20	1361,58	1361,58	3,07	0	1361,58	4,01	0
	30	1025,12	1025,12	5,21	0	1025,12	7,00	0
	40	836,30	839,81	14,97	0,42	838,73	11,43	0,29
	67	524,27	525,90	21,34	0,31	524,27	17,71	0
300	5	15205,3	15218,2	0,53	0,08	15205,3	2,00	0
	10	10404,00	10404	1,59	0	10404,00	3,87	0
	30	5265,75	5292,99	14,20	0,52	5276,21	13,33	0,20
	60	3095,30	3144,9	37,91	1,60	3144,86	23,62	1,60
	100	1980,49	1980,49	52,23	0	2003,99	33,76	1,19
400	5	67035,20	67035,20	0,79	0	67035,20	2,61	0
	10	46454,00	46538,39	1,96	0,18	46454,00	3,56	0
	40	20009,90	20125,67	21,85	0,58	20146,00	16,67	0,68
	80	12003,50	12053,5	54,76	0	12024,27	51,87	1,33
	133	7679,36	7771,91	94,21	1,21	7711,03	87,56	1,19

Continua

[1] *Software* de modelagem e resolução de problemas lineares e não-lineares de otimização.

TABELA 6 - RESULTADOS DOS TESTES COMPUTACIONAIS COMPARADOS COM TEITZ & BART PARA INSTÂNCIAS GERADAS PARA PROBLEMAS DE *P*-MEDIANAS

Conclusão

n	*p*	LINGO	TEITZ & BART			EDBT		
		S_ÓTIMA	S_MELHOR	TEMPO (s)	*Dp* (%)	S_MELHOR	TEMPO (s)	*Dp* (%)
500	5	85382,42	85404,15	1,56	0,03	85382,42	3,42	0
	10	58590,9	58590,9	4,78	0	58590,9	6,42	0
	50	22136,2	22145,83	39,32	0,04	22379,18	31,76	1,10
	100	13301,6	13667,63	104,24	2,75	13598,47	63,99	2,23
	167	8425,24	8578,9	183,54	1,82	8612,28	147,69	2,22
600	5	102849	102849	2,77	0	102849	4,87	0
	10	71249,4	71307,7	6,62	0,08	71249,4	11,00	0
	60	24210,7	24372,92	49,03	0,67	24402,6	76,65	0,79
	120	14364,2	14465,2	172,45	0,70	14647,88	167,75	1,97
	200	9074,87	9175,53	403,28	1,11	9074,87	355,71	0

O algoritmo proposto EDBT apresentou em 60% das instâncias soluções ótimas, superando o algoritmo Teitz & Bart que obteve a solução ótima em aproximadamente 43,33% das instâncias.

Para as 4 (quatro) instâncias, onde n = 100, o algoritmo EDBT obteve 100% de soluções ótimas e o algoritmo Teitz & Bart também atingiu soluções ótimas, com exceção da instância onde p = 33, onde apresentou um desvio igual a 0,47%. Em relação ao tempo computacional, O algoritmo Teitz e Bart apresentou em todas as instâncias valores menores sendo a diferença média aproximadamente igual a 0,04 segundos.

Para as 6 (seis) instâncias, onde n = 200, o algoritmo EDBT obteve em aproximadamente 83,33% das instâncias soluções ótimas, apresentando apenas na instância onde n = 200 e p = 40 um desvio equivalente a 0,29%, superando algoritmo Teitz & Bart que atingiu a solução ótima em aproximadamente 66,67% das instâncias. O algoritmo Teitz & Bart apresentou, na maioria das instâncias, menores tempos computacionais, com exceção de duas instâncias onde p = 40 e p = 67, registrando nesta última a maior diferença aproximadamente igual a 3,63 segundos a favor do algoritmo EDBT.

Para as 5 (cinco) instâncias, onde n = 300, os algoritmos EDBT e Teitz & Bart apresentaram em 2 instâncias soluções ótimas. O maior desvio, igual a 1,60%, deu-se na instância onde p = 60, apresentado por ambos os algoritmos. O algoritmo

Teitz & Bart apresentou, na maioria das instâncias, menores tempos computacionais, com exceção de duas instâncias onde p = 60 e p = 100, registrando nesta última a maior diferença aproximadamente igual a 18,47 segundos a favor do algoritmo EDBT.

Para as 5 (cinco) instâncias, onde n = 400, os algoritmos EDBT e Teitz & Bart apresentaram soluções ótimas em 2 instâncias. O maior desvio, igual a 1,33%, deu-se na instância onde p = 80, apresentado pelo algoritmo EDBT. Em relação ao tempo computacional, registrou-se a maior diferença entre os algoritmos na instância onde p = 133 equivalente a 6,65 segundos a favor do algoritmo EDBT.

Para as 5 (cinco) instâncias, onde n = 500, o algoritmo EDBT obteve 40% de soluções ótimas, apresentando o maior desvio na instância onde p = 100 equivalente a 2,23%. O Algoritmo Teitz & Bart apresentou soluções ótimas em 20% das instâncias neste grupo e o maior desvio também na instância onde p = 100 equivalente a 2,75%. Em relação ao tempo computacional, registrou-se a maior diferença entre os tempos dos algoritmos na instância onde p = 100 equivalente a 40,25 segundos a favor do algoritmo EDBT.

Para as 5 (cinco) instâncias, onde n = 600, o algoritmo EDBT obteve 60% de soluções ótimas superando o algoritmo Teitz e Bart que apresentou soluções ótimas em 20% das instâncias. O maior desvio na instância onde p = 120 equivalente a 1,97% foi registrado pelo algoritmo EDBT. Em relação ao tempo computacional, registrou-se o maior tempo, neste grupo, na instância onde p = 200 equivalente a 423,28 segundos. Nesta mesma instância observa-se a maior diferença entre os tempos computacionais dos algoritmos que foi de 47,57 segundos.

4.2 TESTES E RESULTADOS PARA OS PMC

Para os testes realizados em problemas de máxima cobertura, foram utilizadas 22 (vinte e duas) instâncias, disponíveis em: http://www.lac.inpe.br/~lorena/ArsigIndex.html, denominado neste trabalho como Lorena_MáxCobertura, contendo 324, 500, 708 e 818 vértices. Os resultados do algoritmo híbrido proposto foram comparados com as soluções ótimas obtidas com o auxílio do *software* LINGO 13.0, adotando-se um raio de cobertura igual a 150, e com os resultados obtidos através da heurística Lagrangeana/*Surrogate*,

apresentados por Pereira e Lorena (2001), onde o raio de cobertura assume valores iguais a 800, 1200 e 1600 e o número de instalações *n* varia de 1 (um) até o mínimo necessário para cobrir todos os vértices.

Seguem, na TABELA 7, resultados obtidos pelo EDBT, para problemas de máxima cobertura, com raio de cobertura *S* igual a 150, comparados com as soluções ótimas obtidas com a utilização do *software* LINGO.

TABELA 7 – RESULTADOS DOS TESTES COMPUTACIONAIS COMPARADOS COM O *SOFTWARE* LINGO PARA PROBLEMAS DE MÁXIMA COBERTURA (S=150M)

n	*p*	LINGO			EDBT			
		S_ÓTIMA	C (%)	TEMPO (s)	S_ EDBT	C (%)	D_p (%)	TEMPO (s)
324	20	7302	60,09	247	7302	60,09	0	4
	30	9127	75,11	230	9103	74,91	0,26	8
	40	10443	85,94	256	10382	85,43	0,58	10
	50	11397	93,79	244	11293	92,93	0,91	15
	60	11991	98,68	246	11868	97,66	1,03	17
	80	12152	100,00	244	12150	99,98	0,02	24
500	40	13340	67,69	745	13220	67,08	0,90	23
	50	14773	74,96	725	14647	74,32	0,85	31
	60	15919	80,78	724	15802	80,18	0,73	38
	70	16908	85,80	750	16710	84,79	1,17	43
	80	17749	90,06	751	17470	88,65	1,57	66
	100	18912	95,97	741	18683	94,80	1,21	76
	130	19664	99,78	734	19510	99,00	0,78	92
	167	19707	100,00	734	19706	99,99	0,01	117
708	70	19481	80,53	1670	19151	79,16	1,69	93
	80	20533	84,88	1744	20180	83,42	1,72	109
	90	21449	88,66	1643	21078	87,13	1,73	129
	100	22173	91,65	1680	21849	90,31	1,46	151
	120	23200	95,90	1661	22805	94,27	1,70	191
	140	23861	98,63	1604	23521	97,23	1,42	237
	180	24185	99,97	1766	24110	99,66	0,31	309
	236	24192	100,00	1692	24192	100,00	0	403

O algoritmo EDBT apresentou solução ótima em 2 instâncias, onde *n* =324 com *p* = 20 e *n* = 708 com *p* = 236.

O maior *Dp* apresentado pelo algoritmo proposto foi de 1,73%.

São mostrados a seguir os resultados do método proposto de acordo com o número de vértices.

Para as 6 (seis) instâncias onde n = 324 o algoritmo EDBT apresentou um desvio percentual médio aproximadamente igual a 0,47. O maior desvio, igual a 1,03%, foi registrado na instância onde p = 60 e para p = 20 o algoritmo atingiu a solução ótima. A média dos tempos computacionais foi de 13 segundos, o que equivale aproximadamente a 5,34% da média dos tempos computacionais apresentados com a utilização do *software* Lingo 13.0.

Para as 8 (oito) instâncias onde n = 500 o algoritmo EDBT apresentou um desvio percentual médio aproximadamente igual a 0,90. O maior desvio, igual a 1,57%, foi registrado na instância onde p = 80 e para p = 167 o algoritmo apresentou o menor desvio percentual aproximadamente igual a 0,01. A média dos tempos computacionais foi de 61 segundos, o que equivale aproximadamente a 8,23% da média dos tempos computacionais apresentados pelo *software* Lingo 13.0 para a solução ótima.

Para as 8 (oito) instâncias onde n = 708 o algoritmo EDBT apresentou um desvio percentual médio aproximadamente igual a 1,26. O maior desvio, igual a 1,73%, foi registrado na instância onde p = 90 e para p = 236 o algoritmo atingiu a solução ótima. A média dos tempos computacionais foi de 203 segundos, o que equivale aproximadamente a 12,04% da média dos tempos computacionais apresentados com a utilização do *software* Lingo 13.0.

Para os testes realizados nas instâncias Lorena_MáxCobertura, descritas anteriormente, o algoritmo EDBT apresentou as mesmas soluções que as apresentadas pela heurística Lagrangeana/*Surrogate* (LAG_SUR) [Pereira e Lorena, 2001]. A heurística lagrangeana/*surrogate* foi implementada em C e executada em um microcomputador equipado com um processador Intel Pentium III 733 MHz e 128 MB de RAM.

Desta forma apresentam-se os resultados referentes somente aos tempos computacionais dos algoritmos, não concluindo comparações entre os mesmos já que testados em diferentes máquinas. Tais resultados são apresentados no GRÁFICO 4 a seguir, onde o rótulo do eixo vertical do mesmo é interpretado da seguinte forma: *n_p_S*.

Os dados referentes as soluções, porcentagem de cobertura e tempos computacionais encontram-se no ANEXO 3 deste trabalho.

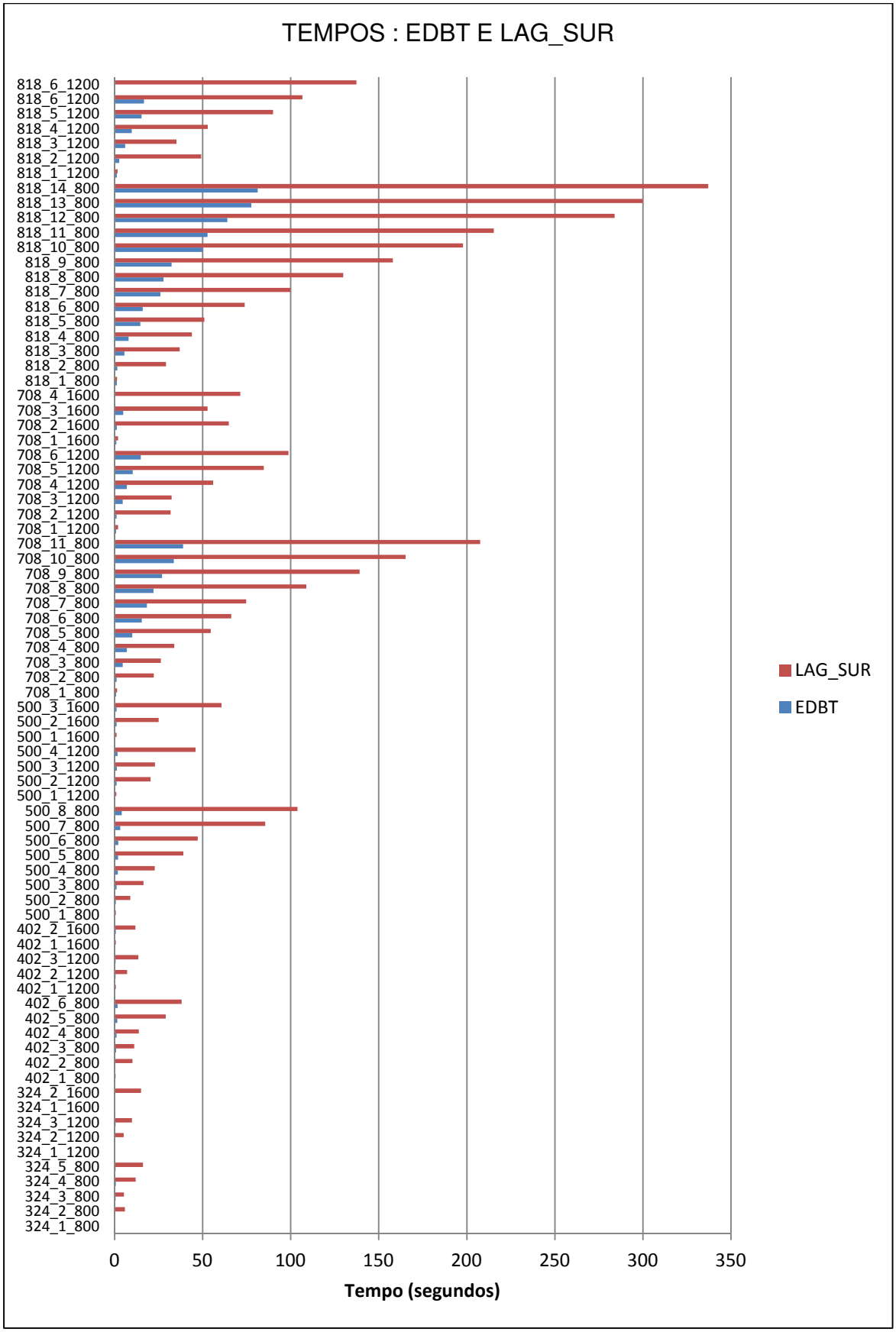

GRÁFICO 4 – TEMPOS COMPUTACIONAIS PARA AS INSTÂNCIAS LORENA_MÁXCOBERTURA.

5 CONCLUSÕES

As metodologias de localização de instalações, baseadas em técnicas determinísticas, requerem altos recursos computacionais (memória de trabalho – RAM, elevados tempos computacionais, entre outros) para atingir boas soluções. Num meio organizacional, onde a distribuição de recursos bem realizada pode garantir melhorias na qualidade do serviço e mesmo contribuir no orçamento da organização, abordar o problema de localização de instalações com métodos exatos não é, geralmente, uma tarefa viável.

Assim, este trabalho apresentou uma nova abordagem heurística híbrida, baseada nos algoritmos de Evolução Diferencial e Busca Tabu, para os problemas de *P*-Medianas e de Máxima Cobertura, visando desenvolver uma proposta de solução que possa realizar a localização de instalações em uma região, dentro de um tempo computacional aceitável. O algoritmo híbrido proposto apresenta uma forma inédita de adaptação do operador de mutação para o algoritmo Evolução Diferencial, desenvolvido inicialmente para problemas onde o espaço solução é contínuo.

Para os problemas de PPM, os testes foram realizados em 50 (cinquenta) instâncias, divididas em dois grupos, cujas soluções utilizadas para comparação foram obtidas através de um método que se utiliza da integração do SIGs ArcView e heurísticas Lagrangeana/*surrogate*, descritas por Lorena *et al.* (2001), as metaheurísticas Algoritmo Genético com Busca Local Adaptativo (AG) baseado no método GRASP, o Algoritmo de Otimização por Saltos de Rãs (JFO), descritos por Vasconcelos *et al.* (2010), o Algoritmo Genético, descrito em Bezerra (2008), o Algoritmo Teitz & Bart e as soluções ótimas do modelo matemático, descrito por Christofides (1975).

Para o grupo de instâncias Lorena, designado para testar problemas de PPM, o algoritmo EDBT obteve em relação aos apresentados por Lorena *et al.* (2001), na maioria das instâncias testadas (70%) melhores soluções. Os tempos computacionais necessários para atingir os melhores resultados foram aceitáveis. Em relação aos resultados apresentados por Vanconcelos *et al.* (2010), O EDBT mostrou em quase todas as instâncias testadas, melhores soluções, com exceção da instâncias em que $n = 818$ e $p = 20$, indicando desta forma, melhorias no que se refere a qualidade de soluções.

Para o grupo de instâncias gerado aleatoriamente, designado para testar problemas de *P*-Medianas e comparado com o algoritmo Teitz & Bart, verifica-se que a média dos desvios percentuais dos dois algoritmos, EDBT igual a 0,49% e Teitz & Bart 0,43%, são muito próximos. Desta forma pode-se perceber que os valores dos desvios percentuais do EDBT são aceitáveis tendo seu máximo equivalente a 2,23% na instância onde $p = 100$ contra o máximo apresentado por Teitz & Bart na mesma instância igual a 2,75%. Observa-se ainda que o algoritmo EDBT apresentou um número maior de vezes a solução ótima em relação ao algoritmo Teitz & Bart.

O algoritmo Teitz & Bart mostrou-se robusto em relação a qualidade de solução, mas observa-se que com o aumento dos dados de entrada o mesmo aumenta também o tempo computacional em relação ao algoritmo EDBT.

Observa-se que para instâncias de pequeno porte, onde *n* varia entre 100 e 400, a obtenção da solução ótima, com a resolução do método exato através do *software* Lingo 13.0, é indicado, pois a diferença entre o tempo computacional com a utilização do *software,* e os apresentados pelos algoritmos EDBT e Teitz & Bart não é representativo. Porém, para as demais instâncias, o EDBT é recomendado, pois processa o algoritmo em tempos computacionais menores, apresentando ainda, desvios percentuais em relação a qualidade solução muito pequenos.

Os problemas de Máxima Cobertura foram resolvidos para 99 (noventa e nove) instâncias, cujas soluções utilizadas para comparação foram obtidas através do software Lingo 13.0, que apresenta as soluções ótimas, e da heurística Langrangeana/*Surrogate* [Pereira e Lorena, 2001]. Verifica-se que a média dos valores percentuais, para as instâncias testadas, está abaixo de 1%, mostrando que as abordagens do EDBT são competitivas para a resolução desses problemas, em tempos computacionais razoáveis.

Problemas de localização de instalações requerem, geralmente, que o algoritmo seja utilizado durante o seu planejamento. Logo o tempo computacional é menos relevante diante da melhoria obtida nas soluções. Como exemplo pode-se citar o planejamento da localização de unidades escolares onde a economia no percurso percorrido é um benefício diário aos usuários destas unidades. Desta forma, o algoritmo EDBT mostrou-se mais eficiente em relação aos métodos concorrentes citados neste trabalho.

Ao comparar os resultados obtidos pelo EDBT com as soluções ótimas, observa-se que o mesmo mostrou-se robusto ao apresentar desvios percentuais

pequenos, dentro de um tempo computacional, que em média representou menos de 11% do tempo obtido com a utilização do *software* Lingo 13.0. Ressalta-se que o algoritmo EDBT apresentou boa qualidade de solução, em tempos computacionais aceitáveis, em instâncias de grande porte (mais de 3000 vértices) onde o *software* não apresentou capacidade operacional para processar a solução.

Verificou-se ainda que o método proposto apresentou menores desvios percentuais em problemas testados, nos quais o número de vértices é maior.

Sendo assim, os resultados obtidos foram satisfatórios, e as soluções encontradas foram consideradas próximas do ótimo. Além disso, o tempo gasto na geração da solução foi consideravelmente baixo, em comparação com os métodos concorrentes e, portanto aceitável para as instâncias abordadas.

5.1 TRABALHOS FUTUROS

Durante a pesquisa realizada neste trabalho foram identificados alguns temas que podem ser objeto de futuras pesquisas que são:

- Investigar o desempenho do algoritmo EDBT para outros problemas combinatórios, como por exemplo, problemas de roteirização, problemas de transporte, problema da mochila, etc;
- Estudar de novas regras determinísticas para os parâmetros visando melhor qualidade das soluções e menor interferência do usuário;
- Considerar um método de inicialização, como por exemplo, a formação de *cluster*s que identifiquem as regiões com maior demanda para a localização inicial das instalações;
- Verificar a aplicabilidade do método em problemas práticos e reais que venham a acrescentar diferentes restrições.

REFERÊNCIAS

AINE, S.; KUMAR, R.; CHAKRABARTI, P. P. **Adaptive Parameter Control of Evolutionary Algorithms to Improve Quality-Time Trade-Off**. Applied Soft Computing, v.9, p.527-540, 2009.

ALMEIDA, L. F.; PACHECO, M. A.; VELLASCO, M. M. **Algoritmos Evolucionários na Otimização de Alternativas para o Desenvolvimento de Campos de Petróleo**, XXIII Encontro Nacional de Eng. De produção, Ouro Preto - MG, Brasil, 2003.

ANDRADE, L. A. C. G.; CUNHA, C. B. **Modelo de Apoio à Decisão para um Problema de Posicionamento de Bases, Alocação e Realocação de Ambulâncias em Centros Urbanos: Estudo de Caso no Município de São Paulo**. Revista Transportes, 2014.

ARANTES, M. B; OLIVEIRA, G. T. S.; SARAMAGO, S. F. P. **Evolução Diferencial Aplicada à Solução de Alguns Problemas de Engenharia de Produção**. FAMAT em Revista, n.6, p.48-61, 2006.

AZZONI, C. R. **Teoria da Localização: uma análise crítica**. São Paulo, IPE - SP, 1982.

BABU, B. V.; JEHAN, M. M. L. **Differential Evolution for Multi-Objective Optimization**. Congress on Evolutionary Computation (CEC '03), p.2696-2703, 2003.

BACK, T.; HAMMEL, U. e SCHWEFEL, H.P. **Evolutionary computation: Comments on the history and current state**. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, 1:3-17, 1997.

BEZERRA, S. N. **Algoritmos evolutivos paralelos aplicados ao problema das p-medianas.** Dissertação de Mestrado, Centro Federal de Educação Tecnológica de Minas Gerais. 2008.

BURKARD, R.E & BONNIGER, T.. **A heuristic for quadratic boolean programs with applications to quadratic assignment problems.** European Journal of Operation Research, vol. 13, 374-386, 1983.

CARRANO, E. G.; TAKAHASHI, R. H. C.; CARDOSO, E. P.; SALDANHA, R. R.; NETO, O.M. **Optimal Substation Location and Energy Distribution Network Design using a Hybrid GA-BFGS Algorithm**. IEE Proceedings-Generation, Transmission and Distribution, v.152, n.6, p.919-926, 2005.

CASTRO, L. N. **Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications**. Danvers: Chapman & Hall/CRC, 2006.

CHAVES, A. A.; **Modelagens Exata e Heurística para Resolução do Problema do Caixeiro Viajante com Coleta de Prêmios**. Universidade Federal de Ouro Preto, Departamento de Computação. Ouro Preto, MG. 2003.

CHURCH, R.; REVELLE, C. **The maximal covering location problem**. *Papers of the Regional Science Association*, v. 32, p. 101–118, 1974.

CORREIA, J.H.; SOUSA FILHO, G.F.; NASCIMENTO, I.Q.; FORMIGA, L.A.C.; NASCIMENTO, R.Q. **Otimização e implementação de um sistema de alocação ótima de serviços públicos utilizando a metaheurística Grasp.** XLII Simpósio Brasileiro de Pesquisa Operacional, Bento Gonçalves, Brasil, Setembro de 2010.

CHRISTOFIDES, N. **Grafh theory – An algorithmic approach**. New York: ed. Academic Press, 1975.

CRAINIC, T.; LAPORTE, G. **Planning models for freight transportation**. European Journal of Operational Research , n. 97, p. 409-438, 1997.

DENG, C.; ZHAO, B.; DENG, A.; LIANG, C. Y. **Hybrid-Coding Binary Differential Evolution Algorithm with Application to 0-1 Knapsack Problems**. International Conference on Computer Science and Software Engineering, p.317-320, 2008.

DENG, C.; ZHAO, B.; YANG, Y.; DENG, A. **Novel Binary Differential Evolution Algorithm for Discrete Optimization**. Fifth International Conference on Natural Computation, p.346-349, 2009.

DUBKE, Alessandra Fraga. **Modelo de localização de terminais especializados: um estudo de caso em corredores de exportação da soja**. Tese de Doutorado em Engenharia Industrial – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2006.

DUMITRESCU, I.; STÜTZLE, T. **Combinations of local search and exact algorithms.** Applications of Evolutionary Computing, LNCS, v. 2611, p. 211-223, 2003.

DUVVURU, N.; SWARUP, K. S. **A Hybrid Interior Point Assisted Differential Evolution Algorithm for Economic Dispatch**. IEEE Transactions on Power Systems, v. 26, n.2, 2011.

EIBEN, A. FREITAS, C.R.; MACHADO, C.M.S.; RETAMOSO, M.R. **Um Método Baseado na Substituição de Vértices e Teoria Espectral para Problemas de *P*-Medianas.** Claio, Simpósio Brasileiro de Pesquisa Operacional, Rio de Janeiro RJ, Brasil, setembro de 2012.

EIBEN, G.; SCHUT, M. C.. **New Ways to Calibrate Evolutionary Algorithms.** Advances in Metaheuristics for Hard Optimization, pp. 153-177, 2008.

EL RHAZI, A.; PIERRE, S.. **A Tabu Search Algorithm for Cluster Building in Wireless Sensor Networks.** Mobile Computing, v. 8, p. 433-444, April, 2009

FERNANDES S.; LOURENÇO, H. R. **Optimised Search Heuristics: A Mapping of procedures and Combinatorial Optimisation Problems**, 2008. Disponível em <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.2032>. Acessado em: 14 mai. 2014.

GAJARDO, A. B.; VILLALOBOS, E. P.; BURGOS, R. O.; MARTÍN, C. O. S. **Algoritmo Basado en Discriminacion por Distancias con Busqueda Global Aplicado al Problema de la *P*-Mediana.** Revista Ingeniería Industrial - Año 9 N° 1: 87 - 94 , Chile, 2010.

GALVÃO, R. D. **Uncapacitated Facility Location Problems: Contributions**. Pesquisa Operacional. v. 24, no 1, 2004.

GAREY, M. R. e JONHSON, D. S. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. W.H. Freeman and Co, 1979.

GENDREAU, M.; HERTZ, A.; LAPORTE,G.. **A Tabu Search Heuristic for the Vehicle Routing Problem.** INFORMS, Management Science, Vo1. 40, No. 10, pp. 1276-1290, October 1994.

GLOVER, F. **Future Paths for Integer Programming and Links to Artificial Intelligence**. *Computers and Operations Research*, 13:533-549, 1986.

GLOVER, F.; LAGUNA, M. ***Tabu Search***. Kluwer Academic Publishers, Boston, 1997.

GOMES, A. **Uma Introdução à Busca Tabu.** Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo –SP, 2009. <http://www.ime.usp.br/~gold/cursos/2009/mac5758/AndreBuscaTabu.pdf>. Acessado em 22/maio/ 2014.

HAKIMI, S.L. **Optimum Location of Switching Centers and the Absolute Centers and the Medians of a Graph.** Operations Research, 12: 450-459, 1964.

HAKIMI, S. **Optimum Distribution of Switching Centers in a Communication Network and Some Related Graph Theoretic Problems.** Operations Research, v. 13, n. 3, p. 462–475, 1965.

HERTZ, A.; TAILLARD, E.; de WERRA, D. **A Tutorial on Tabu Search**. Proc. of Giornate di Lavoro AIRO'95, Entreprise Systems: Management of Technological and Organizational Changes, p. 13-24, 1995.

HIGGINS, A. J. **A Dynamic Tabu Search for Large-Scale Generalized Assignment Problems**. Computers and Operations Research. V. 28, p. 1039-1048, 2001.

HILLIER, F. S.; LIEBERMAN, G. J. **Introdução à Pesquisa Operacional.** 8ª ed. São Paulo: McGraw Hill, 2010.

HORNER, D. **Resolução do Problema das *P*-Medianas não Capacitado: Comparação de Algumas Técnicas Heurísticas.** Tese (Doutorado em Engenharia de Produção), Universidade Federal de Santa Catarina, Florianópolis-SC, 2009.

HOTA, A. R.; PAT, A. **An Adaptive Quantum-Inspired Differential Evolution Algorithm for 0-1 Knapsack Problem**. In: 2010 Second World Congress on Nature and Biologically Inspired Computing, p.703-708, 2010.

JOURDAN, L.; BASSEUR, M.; TALBI, E.G. **Hybridizing Exact Methods and Metaheuristics: A Taxonomy.** European Journal of Operational Research**,** v. 199, n. 3, p. 620-629, 2009.

JUNIOR, R.R.; SANTOS, L.B.L. **Análise Iterativa dos Problemas de *P*-Centros e *P*-Medianas para um Crescente Número de Facilidades: Estudo de Caso na Epidemia de Dengue, Salvador.** 9th Conference on Dynamics, Control and their Applications, Jun 07-11, 2010.

KRAUSE J. **Programação Matemática e Evolução Diferencial para Otimização de Redes de Dutos**. Dissertação. Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, UTFPR, Curitiba-PR, 2014.

KRAUSE J.; LOPES H. **Proposta de um Algoritmo Inspirado em Evolução Diferencial Aplicado ao Problema Multidimensional da Mochila.** Anais do Encontro Nacional de Inteligência Artificial. Curitiba- PR, SBC, 2012.

KRAUSE J.; CORDEIRO J. A.; LOPES H. **Comparação e Métodos de Computação Evolucionária para o Problema da Mochila Multidimensional.** Meta-Heurísticas em Pesquisa Operacional. Editora Omnipax Ltda, Curitiba-PR, 2013a.

KRAUSE J.; PARPINELLI R.; LOPES H. **A Comparison of Differential Evolution Algorithm with Binary and Continuous Encoding for the MKP.** Proceedings of BRICS-CCI. Recife-PE, 2013b

KUEHN, A. A.; HAMBURGER, M. **A Heuristic Program for Locating Warehouses.**

Management Science, 9, p. 643 - 666, 1963.

LIAO, T. W. **Two Hybrid Differential Evolution Algorithms for Engineering Design Optimization.** Applied Soft Computing, v.10, p.1188-1199, 2010.

LIN, C.; QING, A.; FENG, Q. **A Comparative Study of Crossover in Differential Evolution**. Journal of Heuristics, doi: 10.1007/s10732-010-9151-1, publicado online: 18 de novembro de 2010.

LIU, J.; LAMPINEN, J. **A Fuzzy Adaptive Differential Evolution Algorithm**. Soft Computing - A Fusion Foundations and Methodologies and Applications, v.9, n.6, p.448-462, 2005.

LORENA, L. A. N.; SENNE, E. L. F.; PAIVA, J. A. C. e PEREIRA, M. A. **Integração de Modelos de Localização a Sistemas de Informações Geográficas.** Gestão & Produção, v. 8,n. 2, p. 180–195, 2001.

MARANZANA, F. E. **On the Location of Supply Points to Minimize Transportation Costs.** IBM Systems Journal, v. 2, n. 2, p. 129–135, 1964.

MAUTOR, T.; MICHELON, P.. **MIMAUSA: A New Hybrid Method Combining Exact Solution and Local Search**. Proceedings of the 2nd International Conference on Metaheuristics, p. 15. Sophia-Antipolis, France, 1997.

MAUTOR, T.; MICHELON, P.. **MIMAUSA: An Application of Referent Domain Optimization.** Technical Report, 260, Laboratoire d'Informatique, Universit´e d'Avignon et des Pays de Vaucluse, 2001.

NIEVERGELT, J. **Exhaustive Search, Combinatorial Optimization and Enumeration: Exploring the Potential of Raw Computing Power.** SOFSEM: Theory and Practice of Informatics LNCS, p. 18 - 35, 2000.

NOMAN, M.; IBA, H. **Accelerating Differential Evolution Using an Adaptive Local Search**. IEEE Transactions on Evolutionary Computation, v.12, n.1, p.107-125, 2008.

ONWUBOLU, G.; DAVENDRA, D. **Scheduling Flow Shops Using Differential Evolution Algorithm**. European Journal of Operational Research, v.171, p.674-692, 2006.

OSMAN, I. H., LAPORTE, G. **Metaheuristics: a Bibliography**, Annals of Operations Research, vol. 63, pp. 513-623, 1996.

PAMPARÁ, G.; ENGELBRECHT, A. P.; FRANKEN, N. **Binary Differential Evolution**. In: 2006 IEEE Congress on Evolutionary Computation, p.1873-1879, 2006.

PAN, Q. K.; TASGETIREN, M. F.; LIANG, Y. C. **A Discrete Differential Evolution Algorithm for the Permutation Flowshop Scheduling Problem**. Computers & Industrial Engineering, v.55, p.795-816, 2008.

PAN, Q. K.; WANG, L.; QIAN, B. **A Novel Differential Evolution Algorithm for Bi-Criteria No-Wait Flowshop Scheduling Problems**. Computers & Operations Research, v.36, p.2498-2511, 2009.

PEREIRA, M.A.; LORENA, L. A. N. **A Heurística Lagrangeana/Surrogate Aplicada ao Problema de Localização de Máxima Cobertura.** SBPO, XXXIII Simpósio Brasileiro de Pesquisa Operacional,: A Pesquisa Operacional e o Meio Ambiente, Campos de Jordão - SP, novembro 2001.

PRADO, R. S.; SILVA, R. C. P.; GUIMARÃES, F. G.;MAGELA, O. **Uma Nova Abordagem para a Evolução Diferencial em Otimização Discreta.** XVIII Congresso Brasileiro de Automática. Bonito-MS, 2010.

PRICE, K. V.; STORN, R. M. **Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuos Spaces**. Berkeley: Intenational Computer Science Institute, 1995.

PRICE, K. V.; STORN, R. M. **Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuos Spaces**. Journal of Global Optimization, v.11 , p. 341-359, 1997.

PRICE, K. V.; STORN, R. M.; LAMPINEN, J. **Differential Evolution: A Practical Approach to Global Optimization**. Berlin: Springer, 2005.

PUCHINGER, J.; RAIDL, G.R. **Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification.** International Work-Conference on the Interplay Between Natural and Artificial Computation - IWINAC, Part II. LNCS 3562, p. 41–53, 2005.

QIAN, B.; WANG, L.; HUANG, D. X.; WANG, W. L.; WANG, X. **An Effective Hybrid DE-Based Algorithm for Multi-Objective Flow Shop Scheduling with Limited Buffers**. Computers and Operations Research, v.36, n.1, p.209-233, 2009.

REESE, J. **Methods for the *P*-Median Problem: An Annotated Bibliography.** Journal Networks, Vol. 48-3, p. 125-142, New York, USA, 2006.

RIBEIRO, P.C.F., **Um enfoque na Localização de Facilidades Baseado em Testes de Redução e Heurísticas ADD/DROP**. Monografia do Curso da Ciência e Computação da Faculdade Lourenço Filho, Fortaleza – CE, Brasil, 2008.

ROMERO, R.; MONTOVANI, J. R. S. **Introdução a Metaheuristicas**. Anais do III Congresso Temático de Dinâmica e Controle da SBMAC (Sociedade Brasileira de Matemática Aplicada e Computacional), Universidade Estadual Paulista, Ilha Solteira, SP, Brasil, 2004.

ROSÁRIO, R. R. L. **Algoritmos Evolutivos Adaptativos para Problemas de Programação de Pessoal.** Tese. Engenharia de Produção. Universidade Federal de Santa Catarina, SC. 2011.

ROSENKRANTZ, D. E., STREAMS, R. E., LEWIS II, P. M. **An Analysis of Several Heuristics for the Traveling Salesman Problem**, SIAM (Society for Industrial and Applied Mathematics) J. Computer, vol.6, pp.563-581, 1977.

SAUER, J. G. **Abordagem de Evolução Diferencial Híbrida com Busca Local Aplicada ao Problema do Caixeiro Viajante.** Dissertação. Programa de Pós Graduação em Engenharia de Produção e Sistemas. Pontifícia Universidade Católica do Paraná, 2007.

SBALZARINIY, I. F.; MULLER, S. e KOUMOUTSAKOSYZ, P. **Multiobjective Optimization Using Evolutionary Algorithms.** In Proceedings of Center for Turbulence Research Summer Program, pp. 63_74, 2000.

SILIPRANDE, M. D.; CORTES, J. M. R. **Problema de Localização de Antenas de Transmissão para Internet a Rádio no Município de Itaperuna.** XXVIII Encontro Nacional de Engenharia de Produção, Rio de Janeiro-RJ, Brasil, 2008.

SILVA, R. C. P. **Um Estudo sobre a Autoadaptação de Parâmetros na Evolução Diferencial.** Monografia, Departamento de Ciência da Computação. Universidade Federal de Ouro Preto, 2010.

SOUZA, M. J. F. **Programação de Horários em Escolas: Uma Aproximação por Metaheurísticas**. Tese (Doutorado), COPPE/UFRJ, Rio de Janeiro - RJ, Brasil, 2000.

STEFANELLO, F. **Hibridização de Métodos Exatos e Heurísticos para Resolução de Problemas de Otimização Combinatória.** Dissertação da Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Informática, RS, 2011.

SUBRAMANIAN, A.; MEDEIROS, J. M. F.; CABRAL, L. F.; SOUZA, M. F. **Aplicação da Metaheurística Busca Tabu ao Problema de Alocação de Aulas a Salas em uma Instituição Universitária.** Revista Científica Eletrônica de Engenharia de Produção, v. 11, nº 1, UFSC, Florianópolis – SC, Brasil, 2011.

TALBI, E. G. **A Taxonomy of Hybrid Metaheuristics**. Journal of Heuristics, v. 8, n. 5, p. 541-564, 2002.

TASGETIREN, M. F.; SUGANTHAN, P. N.; PAN, Q. K. **An Ensemble of Discrete Differential Evolution Algorithms for Solving the Generalized Traveling Salesman Problem**. Applied Mathematics and Computation, v.215, p.3356-3368, 2010.

TEITZ, M.B.; BART, P. **Heuristics Methods for Estimating the Generalized Vertex Median of a Weighted Grafp.** Operations Research, v. 16, p. 995-961, 1968.

VASCONCELOS, A. M.; SOUZA, S. R.; BEZERRA, S. N. **Aplicação das Metaheurísticas Algoritmo Genético com Busca Adaptativa e Otimização por Saltos de Rãs à Solução do Problema das P-Medianas.** EGENEP, XXX Encontro Nacional de Engenharia de Produção, São Carlos, São Paulo, Brasil, outubro de 2010.

WANG, L.; PAN, Q. K.; SUGANTHAN, P. N.; WANG, W. H.; WANG, Y. M. **A Novel Hybrid Discrete Differential Evolution Algorithm for Blocking Flow Shop Scheduling Problems**. Computers & Operations Research, v.37, p.509-520, 2010.

YEN, J.; LIAO, J. C.; LEE, B.; RANDOLPH, D. **A Hybrid Approach to Modeling Metabolic Systems using a Genetic Algorithm and Simplex Method**. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, v.28, n.2, p.173-191, 1998.

ZAMUDA, A; BREST, J.; BOŠKOVIĆ, B.; ŽUMER, V. **Binary Differential Evolution**. In IEEE Congress on Evolutionary Computation, p.3617-3624, 2007.

ZHANG, M.; ZHAO, S.; WANG, X. **Multi-Objective Evolutionary Algorithm Based on Adaptive Discrete Differential Evolution**. In: 2009 IEEE Congress on Evolutionary Computation (CEC 2009). p.614-621, 2009.

ANEXO 1

Apresentam-se, neste ANEXO 1, os parâmetros adotados para cada instância testada, onde *n* representa a quantidade de vértices do problema, *p* indica a quantidade de medianas a se localizar, $G_{máx}$ número máximo de iterações, N_p a quantidade de vetores da população inicial, δ a quantidade de medianas do V_{melhor} que irão sofrer a mutação e *S* indica, para os problemas de Máxima Cobertura, o raio de cobertura em metros.

A TABELA 8 abaixo indica os parâmetros utilizados nos testes das instâncias Lorena_*P*-Medianas, cujos resultados encontram-se nas TABELAS 3, 4 e 5. Em todos os testes adotou-se *CR* = 0,5.

TABELA 8 – PARÂMETROS PARA AS INSTÂNCIAS LORENA_*P*-MEDIANAS

n	*p*	$G_{máx}$	N_p	δ
324	5	1500	6	2
	10	2000	8	2
	20	2500	8	2
	50	3800	10	2
	108	4000	10	3
818	5	3000	10	2
	10	3000	10	2
	20	3500	10	2
	50	4000	10	3
	100	4500	10	3
	150	5000	12	3
	272	6000	12	3
3282	5	2300	8	2
	10	4000	10	2
	20	4500	10	2
	50	5000	11	2
	100	5500	11	2
	500	10000	12	3
	1000	15000	14	4
	1141	15000	14	4

A TABELA 9, a seguir, indica os parâmetros utilizados nos testes das instâncias geradas aleatoriamente, para PPM. Os resultados computacionais destas instâncias encontram-se adiante na TABELA 6. Para os testes adotou-se $CR = 0{,}5$.

TABELA 9 – PARÂMETROS PARA AS INSTÂNCIAS GERADAS (*P*-MEDIANAS)

n	p	$G_{máx}$	N_p	δ
100	5	800	6	2
	10	1000	6	2
	20	1800	7	2
	33	2200	8	3
200	5	1500	7	2
	10	1600	7	2
	20	1800	7	2
	30	2200	7	2
	40	2300	8	3
	67	2500	8	3
300	5	2000	8	2
	10	2000	8	2
	30	2200	8	3
	60	2800	9	3
	100	3200	10	3
400	5	2400	8	2
	10	2800	8	2
	40	3200	9	3
	80	4000	10	3
	133	4500	10	3
500	5	2800	8	2
	10	3200	9	2
	50	3800	10	3
	100	4200	11	3
	167	4600	12	3
600	5	3200	8	2
	10	5000	9	2
	60	6000	10	3
	120	7000	11	3
	200	7600	11	3

A TABELA 10 abaixo indica os parâmetros utilizados nos testes das instâncias Lorena_MáxCobertura, onde S = 150 metros. Os resultados computacionais destas instâncias encontram-se na TABELA 7. Em todos os testes adotou-se CR = 0,5.

TABELA 10 – PARÂMETROS PARA AS INSTÂNCIAS LORENA_MÁXCOBERTURA COM S=150 METROS

n	p	$G_{máx}$	N_p	δ
324	20	2800	9	2
	30	3000	9	2
	40	3500	2	2
	50	4000	10	2
	60	4000	10	3
	80	4500	10	3
500	40	4000	10	2
	50	4500	10	3
	60	4800	10	3
	70	4800	10	3
	80	5500	12	3
	100	5000	11	3
	130	5500	12	3
	167	5800	12	3
708	70	6000	12	3
	80	6500	12	3
	90	6800	12	3
	100	7500	12	3
	120	8000	12	3
	140	8200	13	3
	180	8800	14	3
	236	9000	15	3

A TABELA 11 abaixo indica os parâmetros utilizados nos testes das instâncias Lorena_MáxCobertura, onde S assume valores iguais a 800, 1200 e 1600 metros. Em todos os testes adotou-se CR = 0,5.

TABELA 11 – PARÂMETROS PARA AS INSTÂNCIAS LORENA_MÁXCOBERTURA COM *S* IGUAL A 800, 1200 E 1600 METROS

n	*p*	*S*	$G_{máx}$	N_p	δ
324	1	800	120	8	1
	2	800	800	10	1
	3	800	1400	10	2
	4	800	1800	8	2
	5	800	1500	8	2
	1	1200	150	8	1
	3	1200	1200	10	2
	1	1600	180	8	1
	2	1600	800	9	1
402	1	800	100	8	1
	2	800	800	10	1
	3	800	1500	10	2
	4	800	2000	10	2
	5	800	2400	10	2
	6	800	2400	10	2
	1	1200	150	10	1
	2	1200	1200	10	1
	3	1200	1200	10	2
	1	1600	200	8	1
	2	1600	1200	10	1
500	1	800	200	8	1
	3	800	1600	10	2
	4	800	2100	11	2
	5	800	2100	11	2
	6	800	2200	11	2
	7	800	3000	12	2
	8	800	3400	12	2
	1	1200	200	8	1
	2	1200	1500	10	1
	3	1200	1800	10	2
	4	1200	2100	11	2
	1	1600	220	8	1
	2	1600	1500	10	1
	3	1600	1800	10	2

ANEXO 2

A TABELA 12 abaixo apresenta os tempos computacionais obtidos para as instâncias da TABELA 6, com a resolução do modelo matemático apresentado por Christofides (1975) através do *software* LINGO 13.0.

TABELA 12 – TEMPOS COMPUTACIONAIS DO *SOFTWARE* LINGO PARA INSTÂNCIAS GERADAS

n	*p*	LINGO	
		S_ÓTIMA	TEMPO
100	5	351,46	18
	10	219,23	15
	20	130,97	16
	33	86,88	15
200	5	3315,98	47
	10	2173,28	39
	20	1361,58	42
	30	1025,12	44
	40	836,3	42
	67	524,27	46
300	5	15205,3	260
	10	10404	256
	30	5265,75	272
	60	3095,3	265
	100	1980,49	259
400	5	67035,20	455
	10	46454	481
	40	20009,9	519
	80	12003,5	521
	133	7679,36	511
500	5	85382,42	901
	10	58590,9	848
	50	22136,2	856
	100	13301,6	886
	167	8425,24	877
600	5	102849	1336
	10	71249,4	1362
	60	24210,7	1311
	120	14364,2	1345
	200	9074,87	1340

ANEXO 3

A TABELA 13 abaixo apresenta resultados computacionais dos métodos EDBT e Langrangeana/*Surrogate*, apresentado por Pereira e Lorena (2010). A comparação entres os tempos computacionais mostram-se no GRÁFICO 4. A legenda SOLUÇÃO indica a melhor solução apresentada por ambos os métodos abordados.

TABELA 13 - RESULTADOS DOS TESTES COMPUTACIONAIS COMPARADOS COM A LANGRANGEANA/*SURROGATE* PARA PROBLEMAS DE MÁXIMA COBERTURA (S = 800, 1200, 1600 METROS)

n	*p*	*S*	SOLUÇÃO	C (%)	TEMPO PEREIRA	TEMPO EDBT
324	1	800	5461	44,94	0,28	0,04
	2	800	8790	72,33	5,92	0,37
	3	800	11694	96,23	5,33	0,62
	4	800	12106	99,62	11,92	0,69
	5	800	12152	100	16,2	0,64
	1	1200	9932	81,73	0,27	0,05
	2	1200	11555	95,09	5,22	0,16
	3	1200	12152	100	9,84	0,52
	1	1600	12123	99,76	0,27	0,06
	2	1600	12152	100	15	0,29
402	1	800	6555	41,01	0,55	0,05
	2	800	11339	70,94	10,16	0,4
	3	800	14690	91,9	11,09	0,86
	4	800	15658	97,96	13,73	1,18
	5	800	15970	99,91	29,11	1,53
	6	800	15984	100	38,01	1,75
	1	1200	10607	66,36	0,71	0,04
	2	1200	14832	92,79	7,14	0,6
	3	1200	15984	100	13,46	0,59
	1	1600	15438	96,58	0,77	0,1
	2	1600	15984	100	11,87	0,69
500	1	800	7944	40,31	0,77	0,14
	2	800	12454	63,2	8,89	0,67
	3	800	15730	79,82	16,42	1,16
	4	800	17794	90,29	22,79	1,81
	5	800	18859	95,7	39,06	1,98
	6	800	19525	99,08	47,18	2,18

continua

TABELA 13 - RESULTADOS DOS TESTES COMPUTACIONAIS COMPARADOS COM A LANGRANGEANA/*SURROGATE* PARA PROBLEMAS DE MÁXIMA COBERTURA (S = 800, 1200, 1600 METROS)

continuação

n	*p*	*S*	SOLUÇÃO	C (%)	TEMPO PEREIRA	TEMPO EDBT
500	7	800	19692	99,92	85,58	3,29
	8	800	19707	100	103,87	4,07
	1	1200	10726	54,43	1,04	0,14
	2	1200	18070	91,69	20,48	1,11
	3	1200	19393	98,41	22,9	1,24
	4	1200	19707	100	45,92	1,68
	1	1600	14804	75,12	1,15	0,16
	2	1600	19668	99,8	25,04	1,1
	3	1600	19707	100	60,74	1,18
708	1	800	8393	34,69	1,48	0,9
	2	800	13306	55	22,25	1,17
	3	800	17272	71,4	26,25	4,58
	4	800	20338	84,07	33,84	6,92
	5	800	21486	88,81	54,65	10,09
	6	800	22504	93,02	66,19	15,46
	7	800	23151	95,7	74,65	18,32
	8	800	23667	97,83	108,81	22,04
	9	800	24024	99,31	139,18	26,94
	10	800	24163	99,88	165,26	33,65
	11	800	24192	100	207,51	38,87
	1	1200	11612	48	1,98	0,93
	2	1200	20376	84,23	31,86	1,21
	3	1200	22422	92,68	32,4	4,66
	4	1200	23884	98,73	55,97	7,02
	5	1200	24142	99,79	84,69	10,33
	6	1200	24192	100	98,7	14,94
	1	1600	16827	69,56	2,04	0,95
	2	1600	23366	96,59	64,87	1,24
	3	1600	23888	98,74	52,73	4,87
	4	1600	24192	100	71,4	7,71
818	1	800	8393	28,77	1,48	1,26
	2	800	13306	45,62	29,16	1,58
	3	800	17507	60,02	37,02	5,59
	4	800	21428	73,46	43,83	7,99
	5	800	24531	84,1	51,03	14,65
	6	800	25908	88,82	73,87	16,06
	7	800	26933	92,34	99,8	25,98
	8	800	27783	95,25	129,84	27,83

continua

TABELA 13 - RESULTADOS DOS TESTES COMPUTACIONAIS COMPARADOS COM A LANGRANGEANA/*SURROGATE* PARA PROBLEMAS DE MÁXIMA COBERTURA (S = 800, 1200, 1600 METROS)

conclusão

n	*p*	*S*	SOLUÇÃO	C (%)	TEMPO PEREIRA	TEMPO EDBT
	9	800	28351	97,2	158,02	32,33
	10	800	28639	98,19	197,79	49,88
	11	800	29019	99,49	215,36	52,72
	12	800	29103	99,78	283,91	64,03
	13	800	29144	99,92	299,89	77,65
	14	800	29168	100	337,02	81,25
818	1	1200	11612	39,81	1,71	1,32
	2	1200	20290	69,56	49,16	2,72
	3	1200	25211	86,43	35,21	6,04
	4	1200	27029	92,67	52,95	9,77
	5	1200	28513	97,75	89,97	15,24
	6	1200	29137	99,89	106,61	16,71
	7	1200	29168	100	137,31	26,34